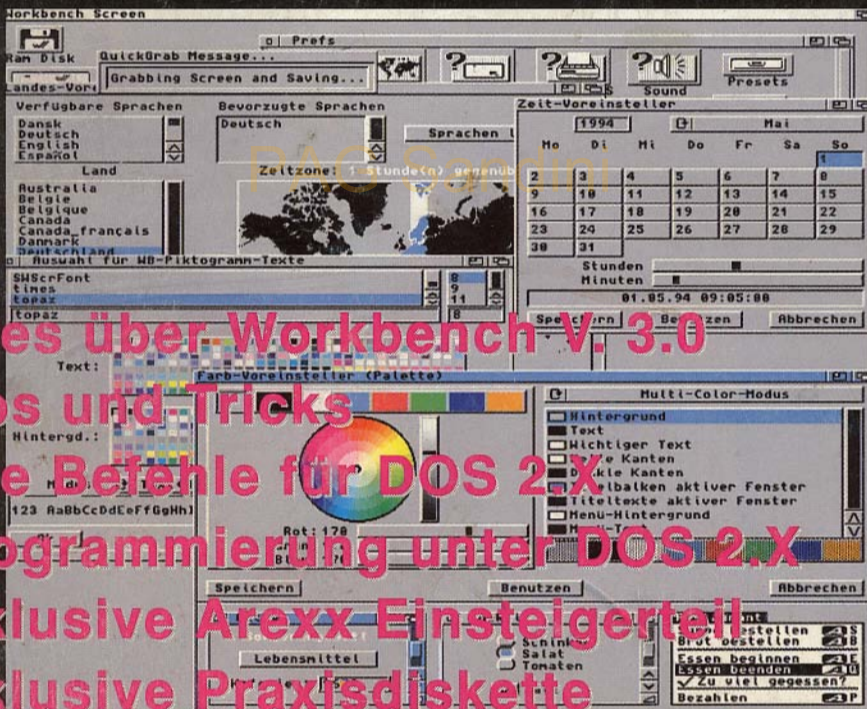


AMIGA
ACHBUCH

Anton Mückl

AMIGA

DOS 2.0/3.0



Alles über Workbench V. 3.0

Tips und Tricks

Alle Befehle für DOS 2.x

Programmierung unter DOS 2.x

Inklusive Arexx Einstiegsleitfaden

Inklusive Praxisdiskette

media
verlagsgesellschaft mbH

AMIGA **DOS 3.0**

PAG Sandini

Autor:
Anton G. Mückl

PAG Sandini



Waldweg 5 • 88175 Scheidegg /Allg.
Tel. 0 83 87/ 80 52 • Fax 0 83 87/83 55

Satz u. Belichtung:

media Verlagsgesellschaft mbH, Scheidegg

© media Verlagsgesellschaft mbH

Alle genannten Produkt/Warenbezeichnungen sind eingetragene Warenzeichen der Hersteller. Alle Rechte vorbehalten. Kein Teil der Anleitung, des Softwareprogrammes darf in irgendeiner Form, ohne ausdrückliche, schriftliche Genehmigung der *media Verlagsgesellschaft mbH* reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Inhalt

Vorwort.....	7
1.0 Einleitung.....	8
2.0 Die Workbench.....	11
2.1 Das Menü	11
2.1.1 Der Menü-Titel „Workbench“.....	13
2.1.2 Der Menü-Titel „Window“	15
2.1.3 Der Menü-Titel „Icons“	18
2.1.4 Der Menü-Titel „Tools“	24
2.2 Die Verzeichnisse der Workbench.....	25
2.2.1 Die Voreinstellungs-Programme.....	26
2.2.1.1 Einstellungen.....	26
2.2.1.2 Time.....	27
2.2.1.3 Input	28
2.2.1.4 Palette	30
2.2.1.5 WBPattern	33
2.2.1.6 Pointer	35
2.2.1.7 Font.....	36
2.2.1.8 Screenmode	37
2.2.1.9 Overscan.....	39
2.2.1.10 Printer	41
2.2.1.11 PrinterGfx	42
2.2.1.12 Serial.....	45
2.2.1.13 I-Control.....	46
2.2.1.14 Sound.....	49
2.2.1.15 PrinterPS.....	50
2.2.1.16 Locale	51
2.2.2 Utilities	53
2.2.2.1 Clock	53
2.2.2.2 More	53
2.2.2.3 Display	55
2.2.2.4 MultiView.....	55
2.2.2.5 Say	56
2.2.2.6 Exchange	56

2.2.3 Tools	56
2.2.3.1 Calculator	57
2.2.3.2 Grafikdump	57
2.2.3.3 CMD	57
2.2.3.4 IconEdit	58
2.2.3.5 MEmacs	60
2.2.3.6 Colors	60
2.2.3.7 InitPrinter	61
2.2.3.8 KeyShow	62
2.2.3.9 PrintFiles	63
2.2.3.10 ShowConfig	64
2.2.4 Der „Commodities“-Ordner	64
2.2.4.1 Exchange	65
2.2.4.2 AutoPoint	66
2.2.4.3 ClickToFront	66
2.2.4.4 Blanker	66
2.2.4.5 NoCapsLock	67
2.2.4.6 I Help	68
2.2.4.7 FKey(WB 2.0)	68
2.2.4.8 FKey(WB 2.1 und 3.0)	69
2.2.4.9 CrossDOS	70
2.2.5 System	70
2.2.5.1 NoFastMem	71
2.2.5.2 SetMap	71
2.2.5.3 DiskCopy	71
2.2.5.4 Format	72
2.2.5.5 RexxMast	73
2.2.5.6 FixFonts	74
2.2.5.7 Fountain	74
2.2.5.8 CLI und Shell	75
2.2.6 Devices	76
2.2.6.1 DosDrivers	77
2.2.6.2 Keymaps	80
2.2.6.3 Monitors	80
2.2.6.4 Printers	81
2.2.6.5 DataTypes	81
2.2.7 Expansion und WBStartup	82
2.2.7.1 WBStartup	82

2.2.7.2 Expansion	83
2.3 CrossDos	85
2.3.1 Namenskonventionen unter MS-DOS	84
2.3.2 CrossDos - Das Commodity	85
3.0 Amiga DOS	87
3.1 Grundlegendes	87
3.2 Shell - die Benutzeroberfläche	87
3.2.1 Editieren in der Shell	88
3.2.2 Die Eigenschaften des Shell-Fensters	92
3.2.3 Die Escape-Sequenzen	94
3.2.4 Programmaufruf in der Shell	98
3.3 Das Verzeichnissystem der Datenträger	101
3.4 Die Jokerzeichen von AmigaDOS	104
3.5 Logische und physikalische Geräte	105
3.6 Die AmigaDOS-Befehle	111
3.6.1 Die Befehlskonventionen und Schablone	111
3.6.2 Ein erster Überblick	115
3.7 MEMACS der Texteditor	455
3.7.1 Der Aufruf von MEMacs	456
3.7.2 Das Menü des MEMacs	459
3.7.2.1 Das Menü „Project“	460
3.7.2.2 Das Menü „Edit“	464
3.7.2.3 Das Menü „Window“	468
3.7.2.4 Das Menü „Move“	470
3.7.2.5 Das Menü „Line“	472
3.7.2.6 Das Menü „Word“	473
3.7.2.7 Das Menü „Search“	474
3.7.2.8 Das Menü „Extras“	476
3.7.3 Weitere MEMacs-Befehle	481
3.7.4 Eigene Voreinstellung für MEMacs	483
3.8 Die MountList	483
3.9 AmigaDos-Fehlermeldungen	489
4.0 System-Fehlermeldungen	503
4.1 Der Aufbau der Fehlernummer	503
4.2 Prozessor-Fehler	504
4.3 Betriebssystem-Fehler	506

4.3.1 Die Fehler der „exec.Library“	507
4.3.2 Die Fehler der „graphics.library“	508
4.3.3 Fehler der „layers.library“	509
4.3.4 Fehler der „intuition.library“	510
4.3.5 Fehler der „dos.library“	511
4.3.6 Fehler der „ram.library“	512
4.3.7 Fehler der „expansion.library“	512
4.3.8 Fehler des „trackdisk.device“	513
4.3.9 Fehler des „timer.device“	513
4.3.10 Fehler der DiskResource.....	513
4.3.11 Fehler des BootStrap.....	513
5.0 A Rexx für Einsteiger	514

PAG Sandini

Vorwort

Als ich voller Stolz die Verpackung des Amiga 1200 ausräumte, wurde meiner Freude ein jähes Ende bereitet. Wo war das Benutzerhandbuch zum "AmigaDOS"? Bei meinen Rückfragen beim Händler und bei Commodore in Frankfurt wurde mir dann nach und nach klar, daß Commodore bewußt dem Anwender des Amiga 1200 dieses Handbuch vorenthalten würde - "aus Kostengründen". Auf der CeBit 1993 wurde ich getröstet, daß bereits fieberhaft an der deutschsprachigen Ausgabe des AmigaDOS-Benutzerhandbuches gearbeitet wurde. Als auch ein ganzes Jahr später immer noch kein Buch auf dem Markt zu finden war, war mir klar, daß dies auch wohl noch länger so sein würde. Gerade als ich meinem Frust richtig Luft verschaffen wollte, hat mir dann die Media Verlagsgesellschaft angeboten, doch das Handbuch selbst zu schreiben, schließlich hätte ich ja genügend Erfahrung mit dem Amiga und speziell den älteren Versionen des AmigaDOS.

Es wäre wohl die beste Art, die Frustration abzubauen, dachte ich und so willigte ich ein. Ich war der Meinung, daß dies wohl nicht allzuviel Arbeit sein würde, und erinnerte mich dabei an die AmigaDOS-Kurse, die ich für die Versionen 1.2 und 1.3 geschrieben hatte. Ich habe dabei wohl die Möglichkeiten, die das neue AmigaDOS der Version 3.0 bietet, gründlich unterschätzt. So wurde das Manuskript des Buches immer länger und länger, den Termin für die Abgabe hatte ich sowieso schon längst verpaßt. Es waren selbst für einen Amiga-Freak wie mich immer wieder neue interessante Möglichkeiten zu finden, so daß es für mich eine enorme Überwindung war, nicht gleich alle Entdeckungen weiter zu erforschen - sonst wäre das Buch noch immer nicht fertig. Die Arbeit daran hat mir sehr viel Spaß bereitet, selbst wenn sie wesentlich umfangreicher wurde als geplant.

Da ich durch die Arbeit an diesem Buch viele Mitmenschen vernachlässigt, genervt oder sogar verärgert habe, möchte ich an dieser Stelle allen mein herzliches Dankeschön für ihr mir entgegengebrachtes Verständnis aussprechen. Besonderer Dank gilt vor allem meiner Familie, die oftmals meine Gesellschaft gewünscht oder meine Unterstützung gebraucht hätte und Herrn Harald Mayer für sein geduldiges Warten auf das Manuskript.

1.0 - Einleitung

Daß sich die Maus als Eingabegerät bei den Computern etabliert hat, müssen sogar die Kritiker heute zugeben, bietet doch die graphische Benutzeroberfläche einen leichten und damit schnellen Einstieg in die Welt des Computers. Auch beim Amiga erleichtert die Maus gerade neuen Amiga-Besitzern die ersten Arbeiten mit dem Computer. Doch daß sich mit der Maus, den Gadgets und dem Menü nicht alles machen läßt, haben offensichtlich auch die Entwickler von Commodore einschen müssen, so wurde das Menü der Workbench um den Punkt "Befehl ausführen" erweitert. Leider ist bei Commodore diese Einsicht nicht in alle Abteilungen im eigenen Haus vorgegangen, so daß an anderer Stelle entschieden wurde, dem Amiga 1200 kein Benutzerhandbuch AmigaDOS mitzugeben, wo die Beschreibung der einzelnen Befehle enthalten gewesen wäre. Der Ein- und Aufsteiger zum AmigaDOS Version 3.0 bekam nicht einmal die Chance, die Shell als den Konkurrenten der Workbench kennen zu lernen. Gewiß, viele hätten sich auch nach eingehenden Tests von Workbench und Shell wohl für die graphische Benutzeroberfläche mit den schönen kleinen Bildchen entschieden, weil die Arbeit sehr viel einfacher ist.

Der erste Teil des Buches ist der Workbench (sowohl im Sinne der graphischen Benutzeroberfläche als auch als Systemdiskette) gewidmet. Sie finden dort in kurz gehaltenen Abschnitten jeweils die wichtigsten Informationen zu den einzelnen Programmen der Workbench-Diskette und vor allem der Voreinstellungsprogramme aus der Schublade "Prefs". Die Shell darf als tastaturorientierte Oberfläche nicht unterschätzt werden, so nimmt die Beschreibung aller AmigaDOS-Befehle der Versionen 2.0, 2.1 und 3.0 den größten Teil des Buches in Anspruch. Dazu vielleicht ein paar Anmerkungen:

Ich habe selbst die Erfahrung gemacht, daß am Anfang wirklich nichts der Maus das Wasser reichen kann. Doch nach und nach konnte ich mit der Workbench meine immer komplexeren Probleme nicht mehr lösen. Ich mußte immer öfter auf die Shell - pardon, damals gab es nur den sogenannten CLI (Command Line Interpreter) - zurückgreifen. Als dann mein System durch die Erweiterung mit einer Festplatte so richtig auf Touren kam, konnte ich mir eine sehr komfortable Arbeitsumgebung mit Boot-Menü und verschiedenen Programmsystemen schaffen, daß die Workbench nur noch in

Ausnahmefällen zum Einsatz kam. Das gesamte Menü, daß aus meiner Sicht sehr komfortabel war, wurde aus einzelnen Befehlsdateien, die ausschließlich AmigaDOS-Befehle enthielten, zusammengesetzt. Da ich ohnehin sehr viel mit der Tastatur arbeitete, trat die Maus gänzlich in den Hintergrund. Auch beim neuen Amiga 1200 verlief die Entwicklung ähnlich - anfangs war die Maus das wichtigste Instrument, aber nach und nach arbeite ich auch hier immer mehr mit dem AmigaDOS, die Möglichkeiten der neuen Version 3.0 sind je ohnehin noch viel umfangreicher, als damals bei meinem guten alten Amiga 500. Wenn Sie dieses Buch und ganz besonders den AmigaDOS-Teil sehr konzentriert lesen und die Befehle gleich ausprobieren (wo immer dies möglich ist, evtl. Warnungen und Hinweise beachten), so werden auch Sie sich ein Bild von der Welt machen, die der Maus ewig verschlossen bleiben wird.

Dieses Buch ist sowohl inhaltlich als auch vom Gesamtaufbau zwar eher als Nachschlagewerk zu verwenden, kann aber auch als normale Lektüre verwendet werden. Es stehen bei einigen Befehlen viele interessante Bemerkungen und Hinweise, die in der offiziellen AmigaDOS-Dokumentation nicht enthalten sind und auf Erfahrungen basieren, die ich durch den häufigen Einsatz von AmigaDOS gemacht habe. Im Abschnitt zum AmigaDOS werden alle Befehle der Workbench-Versionen 2.0, 2.1 und 3.0 detailliert beschrieben, wobei ein besonderes Augenmerk auf die verständliche Erklärung der Wirkung der einzelnen Argumente gelegt wurde. Da das Buch in erster Linie als Nachschlagewerk gedacht ist, sind alle Beschreibungen der Befehle, dessen Arbeitsweise und die Wirkung der einzelnen Argumente deutlich voneinander getrennt, so daß sehr schnell die gewünschte Information gefunden werden sollte.

Für den Autor von Computer-Sachbüchern ist es nicht immer einfach, seine Gedanken so zu formulieren, damit auch ein Anwender, der mit der Computerwelt noch nicht richtig vertraut ist, alles auf das erste Mal versteht. Leider ist es oft unmöglich, Wörter aus dem Fachjargon zu benutzen, weil es entweder keine deutschen Übersetzungen dafür gibt oder diese irreführend wären oder aber viel zu umständlich wären. Die Muttersprache der Computer - und damit auch des Amigas - ist Englisch, auch wenn der Amiga mittlerweile teilweise Deutsch gelernt hat. Aus diesem Grund werden Sie immer wieder auf Begriffe aus der englischen Sprache stoßen, die nicht geläufig sein werden, mit denen sich jedoch ein Amiga-Anwender auseinandersetzen sollte.

Aus diesem Grund habe ich oft den englischen Ausdruck dem deutschen vorgezogen. Es gibt aber noch einen weiteren nicht zu unterschätzenden Vorteil, da Gerade in zwei bestimmten Fällen eine Unterscheidung zweier grundsätzlich verschiedener Dinge möglich wird. Ich denke hier an die Begriffe "Screen" und "Bildschirm" bzw. "Window" und "Fenster".

Während sich die letzten beiden Begriffen auf ein und das selbe Objekt beziehen, das allseits bekannte Fenster (etwa der Workbench oder von Schubladen), so können mit "Screen" und "Bildschirm" zwei unterschiedliche Dinge dirket abgegrenzt werden. So wird das Wort "Bildschirm" zum einen für das Gerät benutzt, das einen Netzschalter besitzt, ziemlich flimmert und oft auch hohe Töne von sich gibt - schlichtweg den Bildschirm, den man anfassen kann. Wird dagegen beim Amiga der Begriff "Screen" verwendet, so ist immer der Bildschirm der Workbench oder anderer Programme gemeint, also die Objekte des Betriebssystems, die mit dem Vorder-/Hintergrund-Symbol ausgestattet sind und in denen die Menüs erscheinen. Wie Sie jedoch schon den beiden obigen Sätzen entnehmen können, wird diese Unterscheidung nicht konsequent eingehalten, so daß auch der Begriff "Bildschirm" bisweilen auch für das Objekt des Amiga-Betriebssystems verwendet wird.

Seit der erste Amiga vorgestellt wurde, wurde immer wieder versucht, die englischen Namen für bestimmte Elemente des Amigas einzudeutschen. Wer sich jedoch schon lange mit dem Amiga beschäftigt, wird merken, daß es am einfachsten ist, die Dinge so zu nennen, wie sie heißen. Damit vermeidet man abenteuerliche Namenskonstruktionen und kann immer sicher gehen, daß der andere Gesprächspartner immer weiß, wovon man spricht. Aus dieser sehr subjektiven Überzeugung heraus habe ich viele Begriffe beim englischen Originalnamen genannt und zusätzlich die deutsche Übersetzung angegeben, wo ich es für nötig gehalten habe.

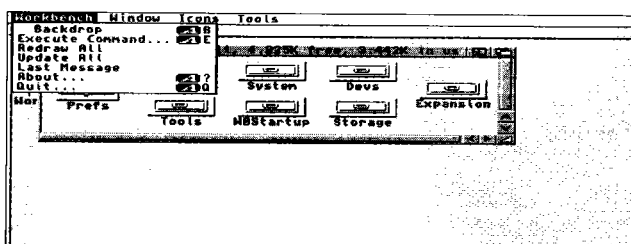
Nach sovielen Begriffen sollten wir uns jetzt jedoch auf die wesentlichen Dinge des Buches konzentrieren. Wenn einige Passagen nicht sofort einleutend sind, empfehle ich Ihnen, den Abschnitt nocheinmal zu wiederholen und auch die genannten Behauptungen durch eigene Experimente zu bestätigen. Durch Ausprobieren und experimentieren kann man mehr lernen, als durch langweiliges lesen der trockenen Materie. Ich wünsche Ihnen viel Spaß beim Arbeiten (und beim Experimentieren) mit dem Amiga und dem AmigaDOS und wünsche Ihnen, daß der Guru ihre Adresse vergessen hat. Gehen Sie auf Entdeckungsreise!

2.0 - Die Workbench

Es gibt zwei sehr unterschiedliche Dinge, die unter Amiga-Benutzern als "Workbench" bezeichnet werden, die Workbench-Diskette (oder das entsprechende Verzeichnis auf einer Festplatte) und die graphische Benutzeroberfläche, das Ding, auf dem man mit dem Mauszeiger so schön herumfahren und die vielen farbigen Bildchen anklicken kann. Damit hier keine Mißverständnisse entstehen, wird in diesem Kapitel beides angesprochen. Dabei werden die wichtigsten Verzeichnisse und die darin enthaltenen Daten und Programme beschrieben, hier beziehen wir uns auf die Diskette hinter der Workbench-Begriff. Vorher sollten wir jedoch die graphische Benutzeroberfläche - vor allem das Menü - etwas näher betrachten.

2.1 - Das Menü

Wenn Sie die rechte Maustaste - die sogenannte "Menütaste" - drücken (und gedrückt halten), verändert sich die oberste Zeile des Bildschirms. Es erscheint am oberen Bildschirmrand die Menüleiste, in der einige Stichworte stehen. Wenn Sie nun den Mauszeiger auf einen dieser Begriffe schieben, klappt ein Menü mit weiteren Stichworten herunter. Nun können Sie einzelne Menüpunkte anfahren, und nachdem Sie den gewünschten ausgewählt haben, lassen Sie die rechte Maustaste wieder los. Dadurch wird ein Menüpunkt angewählt. Bei manchen Menüs klappen weitere Untermenüs auf und Sie können dann aus diesen zusätzliche Menüpunkte auswählen. Die Workbench-Menüs wurden seit der Betriebssystemversion 1.2 ständig weiterentwickelt. Die modernen Menüs bieten nicht nur mehr Funktionen an, sie sind auch logischer gegliedert, als die Menüpunkte der älteren Generationen der Workbench.



Ein typisches Menü

Vor der genaueren Betrachtung noch eine kleine Anmerkung: Viele Menüpunkte sind in "ghostwriting" dargestellt, also in Geisterschrift, oder etwas konkreter ausgedrückt, nicht richtig lesbar. Dies weist meist darauf hin, daß Sie zuerst ein oder mehrere Icons anklicken müssen, bevor die Funktion anwählbar ist. Das geschieht normalerweise mittels eines einzigen Mausklicks. Wenn Sie jedoch gleichzeitig mehr als ein Icon auswählen wollen, müssen Sie die Shift-Taste (für die Großbuchstaben) gedrückt halten und die Icons nacheinander anklicken. Diese Methode kann auch bei den ältesten Amigas angewandt werden. Bei den neueren Amigas ab Kickstart 2.0 und höher, gibt es eine weitere Möglichkeit, mehrere Icons gleichzeitig auszuwählen:

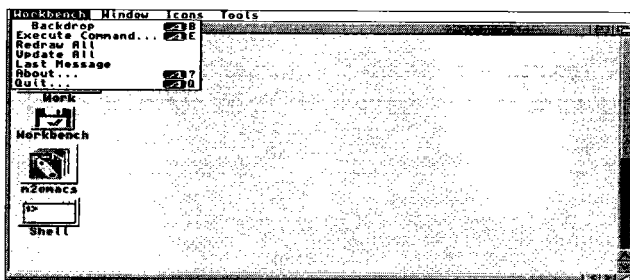
Einfach irgendwo ins Fenster klicken, wo gerade kein Icon oder sonst irgend etwas im Weg ist, und anschließend die Maus (bei gedrückter Maustaste) ziehen. Dann erscheint ein rechteckiger pulsierender Kasten und alle Icons, die sich in ihm befinden, werden ausgewählt, sobald Sie die Maustaste loslassen. Da aber nur rechteckige Ausschnitte möglich sind, wird der Weg zur Tastatur leider doch nicht immer vermeidbar sein.

An dieser Stelle sollte noch darauf hingewiesen werden, daß es nicht nur auf der Workbench (im Sinne der graphischen Benutzeroberfläche) Menüs gibt. Schließlich wäre es Verschwendung, ein derart raffiniertes System nur zu einem einzigen Zweck zu verwenden. Daher werden Sie auch in vielen Anwendungsprogrammen speziell auf den jeweiligen Anwendungszweck zugeschnittene Menüs vorfinden.

Welches Menü letztendlich beim drücken der rechten Maustaste erscheint, hängt jedoch davon ab, welches Fenster im Augenblick aktuell ist. Sollte gerade kein Fenster aktiv sein, so wird immer auf die Workbench zurückgegriffen. Wenngleich die Menüs immer in der Titelleiste des Screens (also der Oberfläche, auf der die Fenster geöffnet werden) erscheinen, so sind diese Menüs dennoch nicht von diesem Screen, sondern vom jeweils aktuellen Fenster abhängig - sofern es sich nicht um Fenster von Schubladen handelt.

Das Menü der Workbench der Versionen 2.0 und 3.0 bietet im Gegensatz zu den älteren Versionen mehr Funktionen, es wurden auch Namen von Funktion geändert und die Struktur überarbeitet. Seit der Kickstart-Version 3.0 gibt es sogar die Möglichkeit, die Namen in deutscher Sprache anzeigen zu lassen. Deswegen wird zu jedem Menüpunkt auch sein deutsches Equivalent angegeben.

2.1.1 - Der Menü-Titel "Workbench"



Backdrop (rechte Amiga-Taste + "B")

Gleich im ersten, dem "Workbench"-Menü (der deutsche Titel ist hier der gleiche) wartet der Menüpunkt: "Backdrop" (übersetzt: "Workbench als Hintergrund"). Denn die Workbench kann nun sowohl als Screen als auch als Window auftreten. Wenn Sie die bekannte Variante (als Screen) bevorzugen, so wählen Sie die Funktion "Backdrop" an, hätten Sie die Workbench jedoch lieber in einem Fenster, so muß dieser Menüpunkte abgewählt werden, d.h. es darf kein Haken neben dem Text stehen.

Der Vorteil der Window-Variante ist natürlich, daß alle Disk-Icons immer leicht zugänglich sind, da Sie ja das Window problemlos in den Vordergrund holen können. Bei einem Workbench-Screen müßten alle Fenster zur Seite geschoben werden. Um Ihre Wahl dauerhaft zu speichern, sollten Sie noch im "Window"-Menü (in der deutschen Version entsprechend "Fenster"-Menü) den Punkt "Snapshot- All" ("fixieren - alles") aufrufen.

Execute Command (rechte Amiga-Taste + "E")

Bei Anwahl von "Execute Command..." ("Befehl ausführen") erscheint ein kleines Fenster, in dem Sie den Namen eines Programmes eintippen können, das Sie ausführen möchten. Dadurch braucht auch der reine Workbench-Benutzer nicht extra eine Shell öffnen, wenn er ein Shell-Kommando starten will. Falls eine Ausgabe stattfindet, so wird ein Fenster geöffnet und der entsprechende Text darin angezeigt. Anschließend müssen Sie dieses Fenster

mit dem Close-Gadget links oben wieder schließen. Weitere Kommandos können nicht unmittelbar ausgeführt werden, aber wie gesagt, für einen einzigen Befehl ist diese Funktion durchaus sinnvoll.

Redraw All

“Redraw All” (zu deutsch “Bild neu aufbauen”) ist nur dann nötig, wenn ein fehlerhaftes Programm die Workbench-Grafik verunstaltet hat. Denn diese Funktion veranlaßt analog der Funktion “Redraw” der Workbench 1.2/1.3 die Workbench, den Bildschirm nochmals komplett neu aufzubauen. Wenn das schuldige Programm neben der Grafik auch noch Programm-Speicher überschrieben haben sollte, werden Sie natürlich dennoch früher oder später nach Indien reisen (Audienz beim Guru!), doch bei ein wenig Grafik-Müll haben Sie gute Chancen, ohne Verzug weiterarbeiten zu können.

Update All

Der Menü-Punkt “Update All” (in der deutschen Version “alles aktualisieren” betitelt) klingt zwar so ähnlich, hat jedoch eine ganz andere Funktion. Es werden nämlich nicht nur alle Fenster neu aufgebaut, sondern auch die zugehörigen Verzeichnisse erneut gelesen. Denn wenn Sie beispielsweise ein Programm und sein Icon über die Shell aus einem Verzeichnis löschen, dessen Window gerade auf der Workbench angezeigt wird, so erfährt die Workbench davon nichts. Mit anderen Worten: Sie sehen möglicherweise Dinge angezeigt, die überhaupt nicht mehr existieren. “Update All” behebt das Problem, indem die Daten aktualisiert werden (nicht nur ihre graphische Darstellung). Wenn Sie natürlich immer auf der Workbench bleiben und auch Ihre Anwenderprogramme die Workbench nicht hinterrücks überrumpeln, werden Sie diese Funktion nicht benötigen, da dann ja sowieso alles seine Ordnung hat.

Last Message

Die Funktion “Last Message” (“letzter Meldung anzeigen”) ist identisch zu der gleichnamigen Funktion der Workbench 1.3, sie dient der erneuten Anzeige einer evtl. aufgetretenen Fehlermeldung in der Titelzeile, was immer dann sinnvoll ist, wenn man diese Meldung nicht mitbekommen hat, weil z.B. ein Fenster die Anzeige verdeckte.

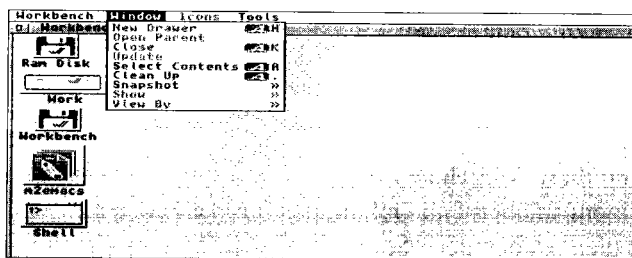
About (rechte Amiga-Taste + “?”)

Auch der Menü-Punkt “About” (deutsch “Version, Copyright ...”, ab der Version 3.0 stehen dahinter auch im englischen Originaltext drei Punkte) darf natürlich nicht fehlen, damit Sie sich immer vergewissern können, welche Kickstart- und Workbench-Version Sie gerade benutzen.

Quit (rechte Amiga-Taste + “Q”)

Und schließlich fehlt noch “Quit” (deutscher Name: “verlassen ...”) - der Menü-Punkt, der die Workbench über die Klinge springen läßt (natürlich nach vorheriger Bestätigung). Seien Sie jedoch gewarnt! Wenn Sie kein Shell- bzw. CLI-Fenster geöffnet haben und auch kein Anwenderprogramm, daß die Möglichkeit bietet, andere Programme zu starten, dann haben Sie den Ast, auf dem Sie saßen, abgesägt. Denn wenn Sie keine Programme mehr starten können, bleibt Ihnen nicht viel mehr übrig, als den Rechner durch einen Neustart wieder benutzbar zu machen.

PAG Sandini 2.1.2 - Der Menü-Titel “Window”



Die deutsche Übersetzung dieses Menü-Titels lautet sinnigerweise “Fenster”, denn die dahinter versteckten Funktionen haben alle mehr oder weniger viel mit den Fenstern auf der Workbench zu tun.

New Drawer (rechte Amiga-Taste + “N”)

Das, was “New Drawer” (oder “Neue Schublade”) im “Window”-Menü leistet, hat der geplagte Anwender der alten Workbench sehr stark vermißt: die

Möglichkeit, eine neue Schublade zu erzeugen. Diese mußten nämlich das standardmäßig vorhandene "Empty"-Verzeichnis duplizieren und anschließend umbenennen. Das hat jedoch seit Version 2.0 ein Ende. Sie wählen einfach das Window aus, in dem Sie gerne das neue Unterverzeichnis erstellt haben möchten und gehen auf "New Drawer". Es erscheint sofort eine neue Schublade mit dem Namen "Unnamed1" und gleichzeitig ein Fenster, in dem Sie diesen Namen ändern können.

Open Parent

Ähnlich anwenderfreundlich ist die Funktion "Open Parent" (der deutsche Titel "übergeordnete Schublade" ist nicht so aussagekräftig). Wenn Sie sich einige Stufen tief in die Verzeichnisstruktur bewegt und alle vorhergehenden Fenster wieder geschlossen haben, um Platz zu schaffen und nun doch noch etwas aus dem übergeordneten Verzeichnis benötigen, so wählen Sie einfach diese Funktion. Bei der Workbench der Version 1.2/1.3 mußte man alle Verzeichnisse ein zweites Mal öffnen. Auf diese Weise können Sie sich natürlich auch Schritt für Schritt wieder nach oben arbeiten, wenn Sie so schneller zum Ziel kommen.

Close (rechte Amiga-Taste + "K")

Einfach zu verstehen, aber im Grunde überflüssig ist "Close" ("schließen"), da Sie Windows weitaus schneller durch das Close-Gadget schließen können, als erst die Close-Funktion des Menüs anzuwählen. Lediglich wenn das Fenster so sehr überlagert ist, daß Sie zwar irgendwo einen kleinen Teil davon sehen können, aber nicht das Close-Gadget und das Vorder-Hintergrund-Symbol, kann diese Vorgehensweise schneller zum Ziel führen. Jedoch laufen Sie in einem solchen Fall Gefahr, das falsche Fenster zu schließen. Wer sich jedoch die sogenannten "Short-Cuts", die Tastaturkürzel der einzelnen Menüpunkte merken kann, könnte jedoch aus diesem Menüpunkt Profit schlagen.

Update (rechte Amiga-Taste + "M"); nur bei deutscher Bezeichnung gültig)

Der Menüpunkt "Update" ("aktualisieren") ist von der Funktionsweise fast identisch mit der Funktion "Update All" aus dem "Workbench"-Menü, es

werden jedoch nicht alle Daten erneut eingelesen sondern nur die aus dem ausgewählten Fenster.

Select Contents (rechte Amiga-Taste + "A")

Die Funktion "Select Contents" (sie wurde mit "alles auswählen" übersetzt) wählt alle Icons aus dem aktiven Fenster aus und erspart es Ihnen, mittels der erweiterten Auswahl jedes Icon von Hand anzuklicken.

Clean Up (rechte Amiga-Taste + ".") **(bzw. "M" bei deutscher Bezeichnung)**

"Clean Up" ("Inhalt aufräumen") ist die ideale Funktion für Leute, die es eilig haben. Sie räumt den Inhalt des gewählten Fensters auf, indem die Icons in einem sauberen Raster platziert werden und zwar abhängig von der Größe dieser Icons. Das Herumschieben von Hand entfällt. Die gleiche Funktion hat auch die ältere Workbench 1.2/1.3 unter dem gleichen Namen im Menü "Special" schon angeboten.

PAG Sandini

Snapshot

Mit "Snapshot" ("fixieren") können Sie einige Informationen bezüglich des aktiven Fensters speichern. Das Untermenü "Window" (deutscher Titel: "des Fensters") fixiert die Größe, Position und die "Show"- und "View by"- Modes (siehe weiter unten). Mit dem Untermenü "All" ("alles") speichern Sie zudem die Positionen aller Icons, die Sie vorher entweder mit "Clean Up" oder von Hand angeordnet haben können. Wenn Sie das Fenster das nächste Mal öffnen, wird alles wieder auf seinem Platz sein.

Show

Wieder eine neue Funktion ist "Show" ("Inhalt anzeigen"). Da üblicherweise nicht alle Programme mit Icons ausgestattet sind, Sie aber vielleicht auf diese Zugriff haben möchten, ohne vorher ins Shell zu müssen, können Sie durch Auswahl von "All Files" (oder im deutschsprachigen Menü durch "alle Dateien") auch diesen Icons zuordnen. Dabei wird selbstständig entschieden, ob es sich um ein Verzeichnis oder ein File handelt und das zugehörige Symbol angezeigt. Die Programme können dann ganz einfach mittels

Doppelklick gestartet werden, was genau den gleichen Effekt hat, wie wenn Sie auf "Execute Command" gegangen wären und den Pfad- und Filenamen eingegeben hätten. Um zur normalen Darstellung zurückzukommen, können Sie dann "Only Icons" ("nur Dateien mit Piktogr.") selektieren, woraufhin wieder nur die tatsächlich vorhandenen Icons angezeigt werden.

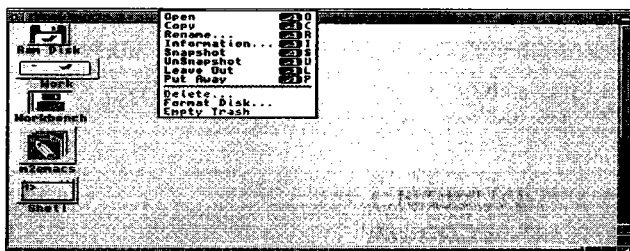
View By

Mit "View By" ("Inhalt auflisten") nähern Sie sich dem CLI oder der Shell sogar noch einen weiteren Schritt. Neben "Icons" ("als Piktogramme"), der Darstellung, die Sie gewöhnt sind, kann der Inhalt eines Verzeichnisses nun auch in Textform angezeigt werden, so wie man es vom "List"-Befehl der Shell kennt. Dabei stehen Ihnen drei Sortierkriterien zur Auswahl: Bei "Name" ("nach Name") werden die Dateien alphabetisch nach dem Namen geordnet, wenn Sie "Date" (oder "nach Datum") auswählen, wird das Erstellungsdatum als Sortierkriterium verwendet und bei "Size" ("nach Größe") wie der deutsche Titel schon erkennen läßt eben nach der Dateigröße. In Verbindung mit "Show" bekommen Sie schon fast alle wirklich vorhandenen Dateien angezeigt. Die Einschränkung "fast" wurde deswegen gemacht, weil die ".info"-Dateien nachwievor herausgefiltert werden, diese enthalten die Daten für die Icons. Doch von der Workbench aus sind diese ja nicht weiter von Interesse.

2.1.3 - Der Menü-Titel "Icons"

Open (rechte Amiga-Taste + "O")

Die erste Funktion des mit "Piktogramm" übersetzten Menü-Titels ist "Open" oder "öffnen". Mit ihm öffnen Sie ein Fenster bzw. starten ein Programm. Da das gleiche mit einem Doppelklick auf das jeweilige Symbol jedoch viel einfacher und schneller erledigt werden kann, werden Sie diese Funktion normalerweise nie benutzen.



Copy (rechte Amiga-Taste + "C")

"Copy" ("kopieren") ist eine interessante Funktion, die mehr in sich hat, als man es von der Workbench 1.3 gewöhnt war. Prinzipiell stehen Ihnen ja zwei Möglichkeiten zur Verfügung, Dateien, Verzeichnisse oder Disketten zu kopieren: Sie ziehen das entsprechende Icon in das gewünschte Fenster bzw. auf das Ziellaufwerk. Oder aber Sie rufen "Copy" auf. Der Unterschied besteht darin, daß Sie mit der ersten Funktion immer von einem Verzeichnis in ein anderes kopieren, bei der zweiten aber eine Kopie im gleichen Verzeichnis erstellen, die dann "Copy_of_" und dem jeweiligen Namen genannt wird. Dies ist immer dann sinnvoll, wenn Sie eine Sicherheitskopie auf der gleichen Einheit erstellen wollen, oder beispielsweise eine Datei verändern möchten, aber dennoch das Original behalten. Dann können Sie später immer wieder auf die "Copy_of_"-Datei zugreifen.

Doch halt, was war das mit den Disketten? Ja, genau, bei der Workbench 2.0/3.0 gibt es keinen "Diskcopy"-Befehl mehr, diese Funktion wird ebenfalls von "Copy" übernommen. Dazu müssen Sie lediglich das Disketten-Icon anwählen und anschließend "Copy". Daraufhin wird diese Diskette über das Laufwerk, in dem sie sich befand, kopiert und Sie regelmäßig zum Wechseln von Quell- und Zieldiskette aufgefordert (sofern nicht ausreichend Speicher vorhanden ist, damit der Kopiervorgang mit einem einzigen Diskettenaustausch auskommt). Wenn Sie über zwei Laufwerke verfügen, können Sie natürlich auch direkt von dem einen auf das andere kopieren. Hierzu ziehen Sie das eine Disketten-Symbol auf das andere.

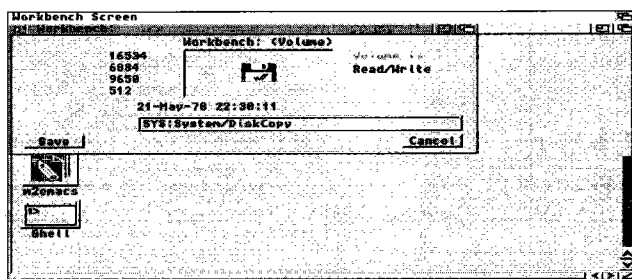
Rename (rechte Amiga-Taste + "R")

Die Funktion "Rename" ("umbenennen ...") macht genau das, was der Name schon sagt: es werden Dateien, Verzeichnisse und Disketten umbenannt.

Dazu müssen Sie jedoch vorher das Symbol der Diskette, Datei oder des Verzeichnisses anklicken, dessen Namen Sie ändern möchten. Anschließend tut der Aufruf dieses Menüpunktes sein übriges: Sie werden in einem Fenster aufgefordert, den neuen Namen einzugeben.

Information (rechte Amiga-Taste + "I")

Um Programmen Parameter zu übergeben, benötigen Sie den Menü-Punkt "Information" ("Informationen ..."). Wird diese Funktion bei angewähltem Icon aufgerufen, so erscheint ein Fenster, in dem nähere Informationen zu dem zugehörigen Programm bzw. Verzeichnis enthalten sind. Es wird das Icon nochmals angezeigt, die Größe, die Anzahl der Blöcke, die es auf dem Datenträger benötigt und wann die letzte Änderung vorgenommen wurde. Darüber hinaus sind aber auch Felder vorhanden, die Sie verändern können. Da wäre zunächst einmal das "Stack"-Feld, in dem die Größe des Stapelspeichers eingegeben werden kann. Im Normalfall sollten Sie diesen Wert nicht ändern, nur bei wenigen Programmen, die ausdrücklich darauf hinweisen, kann es unter Umständen nötig werden, den Wert zu erhöhen. Unter 4096 sollte er jedoch niemals gewählt werden. Weiterhin können Sie noch einen "Kommentar" oder "Comment" zu dem File angeben.



"Information3.0"

Diese Information läßt sich dann auch von der Shell aus mittels "Filenote" abfragen bzw. ändern.

Und um auch den "protect"-Befehl dem Workbench-Anwendern zugänglich zu machen, sind rechts oben sechs Gadget, die den Status der Flags bestimmen. Neben den bekannten ("Readable" = "Lesbar", "Writable" = "Schreibbar", "Executeable" = "Ausführbar" und "Deletable" = "Löschbar")

sind mit der Version 2.0 noch zwei für AmigaDOS neue Flags hinzugekommen:

“Skript” bedeutet, daß ein File eine Script-Datei ist, woraufhin diese nicht mehr mit dem Befehl “Execute” gestartet werden muß, sondern wie jedes andere Programm von der Shell aus gestartet werden kann, indem lediglich der Dateiname eingegeben wird.

“Archived” (“Archiviert”) schließlich ist ein Flag, daß bei jedem schreiben den Zugriff auf die Datei automatisch gelöscht wird. Dadurch ist es leicht feststellbar, ob sich eine Datei verändert hat. Dies ist besonders für Backup-Programme sehr hilfreich, da dann nur die veränderten Dateien gesichert werden müssen.

Doch die wichtigste und auch schon angesprochene Funktion ist die der Parameterübergabe beim Aufruf von Programmen. Dazu gibt es das “Tool Types”-Feld (in der deutschen Version wird das Feld mit “Merkmale” benannt), in dem Sie die gewünschten Parameter eingeben können. Das Format ist üblicherweise “Schlüsselwort = Argument”. Welche Schlüsselwörter Ihnen zur Verfügung stehen, ist bei der Workbench 2.0/3.0 bereits in dem “Tool Types”-Feld angegeben und zwar in spitzen Klammern. Das bedeutet, daß diese Angaben überlesen werden. Bei älteren Programmen ist dies oftmals nicht der Fall, dann sind die Parameter aber in der Dokumentation zu dem Programm erklärt.

Wollen Sie nun einen neuen Parameter eingeben, so klicken Sie “New” (oder “Neu”) an und können anschließend in dem Text-Feld, in dem sich nun der Cursor befindet, die Parameterzeile nach obigen Format eingeben. Um eine bereits bestehende Zeile zu verändern, genügt es, diese anzuklicken, woraufhin Sie diese dann im unteren Textfeld editieren können. Sie können auch Zeilen wieder löschen, indem Sie diese zuerst anwählen und dann auf “Del” (oder “Löschen”) klicken. Jegliche Änderung sollten Sie nach der Bearbeitung durch Klick auf das “Save”-Gadget abspeichern.

Snapshot (rechte Amiga-Taste + “S”)

Die Funktion “Snapshot” (übersetzt “fixieren”) macht für einzelne Icons genau das, was die gleichnamige Funktion aus dem “Window”-Menü für Fenster macht: Sie speichert die Position von Symbolen in einem Fenster.

Dadurch erscheint es beim Öffnen des Fensters seines Verzeichnisses immer an der vorher bestimmten Stelle. Im allgemeinen wird man zwar alle Icons so positionieren, wie man es sich vorstellt und dann die Einstellungen in einem Rutsch abspeichern, doch gerade im Diskettenbetrieb kann das doch etwas dauern. Wenn Sie also nur ein neu hinzugekommenes Icon fixieren wollen, so ist diese Funktion sicherlich effektiver.

Unsnapshot (rechte Amiga-Taste + "U")

"Unsnapshot" (dt: "Position freigeben") hebt diese Einstellungen wieder auf, die Workbench plazierte die Icons dann nach eigenem Ermessen. Dies ist besonders dann sehr hilfreich, wenn Sie in einer Schublade sehr oft Dateien legen und andere wieder entfernen. Denn in diesem Fall ist es einfacher, es der Workbench zu überlassen, die Icons halbwegs sinnvoll zu plazieren, als jedesmal dem neuen Zustand entsprechend alle Positionen erneut zu fixieren.

Leave Out (rechte Amiga-Taste + "L")

Eine weitere Neuheit - sicherlich eine große Verbesserung - bringt die Funktion "Leave Out" oder "auslagern". Da häufig benötigte Programme grundsätzlich am tiefsten in der Verzeichnisstruktur zu finden sind (Murphy's Gesetz), müssen Sie sich vor deren Start immer erst durch eine Unmenge von Windows wühlen. Viel einfacher wäre es doch, das Icon direkt auf der Workbench zu haben. Und genau dazu ist "Leave Out" da. Das angewählte Icon wird auf die Workbench gelegt, wenngleich sich die Position des Files selbst nicht verändert. Aber Sie können immer sofort auf das Programm zugreifen, selbst nach einem Neustart.

Put Away (rechte Amiga-Taste + "P")

Wenn Sie das Icon wieder auf seinen eigentlichen Platz zurücklegen möchten, bedienen Sie sich des Menüpunkts "Put Away" oder "zurücklegen". Das Icon verschwindet von der Workbench und taucht in seinem ursprünglichen Fenster wieder auf.

Delete

"Delete" oder "löschen ..." ist eine jener Funktionen, die wieder einmal genau das tut, was man von ihr erwartet. Es wird das angewählte Programm - es kön-

nen aber auch Verzeichnisse sein - gelöscht! Zwar erst nach einer Rückfrage, dann aber unwiederruflich. Es gibt ja auch eine sichere Möglichkeit, Dateien und Verzeichnisse zu entfernen, bei der der entgültig letzte Schritt nicht sofort ausgeführt werden braucht. Dazu brauchen Sie nur die Programme in den Papierkorb kopieren. Erst wenn Sie diesen ausleeren, sind die darin enthaltenen Daten verloren (dazu jedoch weiter unten mehr).

Format Disk

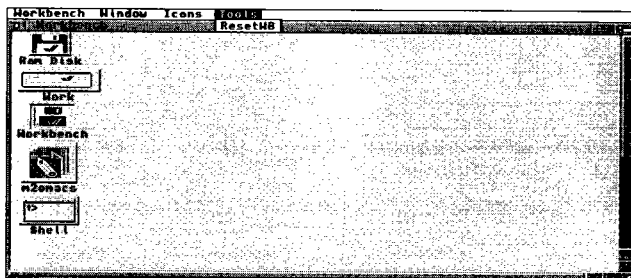
“Format Disk” oder auch “Disk formatieren” ist nur anwählbar, wenn Sie vorher auf ein Diskettensymbol geklickt haben. Anschließend ist es möglich diese Diskette zu formatieren, d.h. komplett zu löschen. Bei einer neuen Diskette, oder einer solchen, die in einem fremden Format bespielt war, müssen Sie bei der folgenden Sicherheitsabfrage “OK” anwählen - bei der Workbench 3.0 heißt das Gadget “Format” oder “Formatieren”. Nur wenn die Diskette bereits im normalen AmigaDOS-Format bespielt war, sollten Sie “OK-Quick” bzw. “Quick Format” oder “Schnell” anklicken, da bei dieser Variante lediglich der Root-Block der Diskette gelöscht wird, was zwar sehr viel schneller geht, allerdings bei einem Zugriff auf einen uninitialisierten Block Fehler verursacht.

Diese Funktion löscht bei schnellem formatieren nicht die Daten direkt, sondern gibt nur den von ihnen belegten Speicherplatz zum wiederbeschreiben frei. Dadurch können einige Programme die Daten wieder herstellen, während bei der vollständigen Formatierung nichts mehr zu machen ist. Verlassen Sie sich jedoch nicht auf diese Programme, auch die können einmal Fehler machen und keine Daten mehr hervorholen.

Empty Trash

Der letzte Punkt dieses Menüs ist “Empty Trash” oder “Papierkorb leeren”. Diese Funktion kann nur dann aufgerufen werden, wenn der Papierkorb angewählt wurde. Es werden dann die Files, die Sie irgendwann in den Papierkorb geworfen haben, endgültig gelöscht. Damit sind Sie ebenso verloren, wie wenn Sie gleich “Delete” angewählt hätten; nur solange die Dateien noch im Papierkorb lagen, hätten Sie sie wieder herausholen können.

2.1.4 - Der Menü-Titel "Tools"



"Menü 3.0-4"

Diese Menü, das in der deutschen Version "Hilfsmittel" benannt ist, ist nicht für die Workbench selbst, sondern vielmehr für Anwenderprogramme reserviert. Falls eines Ihrer Programme also diese Funktion unterstützt, so können Sie hier einen neuen Menüpunkt installieren und das Programm dann über diesen starten, anstatt das Icon anzuklicken.

Wie Sie zu diesem Zweck vorzugehen haben, ist von Anwendung zu Anwendung unterschiedlich; ziehen Sie diesbezüglich die Dokumentation des Programms zu Rate. Einen Punkt weist dieses Menü aber doch schon auf, nämlich "ResetWB" oder "WB rücksetzen". Was man mit ihm anstellt, dürfte wohl intuitiv klar sein: die Workbench zurücksetzen. "Zurücksetzen" bedeutet dabei, daß sämtliche Windows geschlossen werden, die Workbench in ihren Ausgangszustand versetzt wird und die vorher vorhandenen Fenster wieder geöffnet werden.

Wann Sie jemals in die Verlegenheit kommen sollten, diesen Menüpunkt aufrufen zu müssen, ist unklar. Sollten aber wirklich einmal seltsame Fehler auf der Workbench auftreten, kann es auf keinen Fall schaden, diese Funktion anzuwählen - wenngleich die Erfolgchancen sehr gering sind. Aber einen Versuch ist es wert.

2.2 - Die Verzeichnisse der Workbench

Wer unter dem Begriff "Workbench" nicht an die graphische Benutzeroberfläche sondern an die beim Amiga mitgelieferten System-Disketten gedacht hat, kommt jetzt auf seine Kosten. Natürlich hat sich auch die Workbench-Diskette seit der ersten Version stark verändert. Während am Anfang noch alle Verzeichnisse auf einer Diskette Platz fanden - es war sogar noch Platz für ein paar kleine Demo-Programme -, wurden zunehmend mehr Daten und auch Programme auf andere Disketten ausgelagert. Ab der Version 3.0 sind beispielsweise nicht einmal mehr die Voreinstellungs-Programme vollständig auf der Workbenchdiskette zu finden. Doch selbst für Aufsteiger von älteren Amiga-Modellen sind diese Veränderungen jedoch auf den ersten Blick nicht zu erkennen, da die ausgelagerten Daten und Verzeichnisse keine Icons besitzen und daher auch vorher nicht auf der Workbench zu sehen waren. Von den mit Symbolen versehenen Verzeichnissen sind nachwievor die Directories "System", "Utilities", "Expansion", "Empty" sowie in der Regel auch der Trashcan als Sonderverzeichnis vorhanden.

Seit der Version 1.3 existiert das Verzeichnis "Prefs". Das Programm zum Festlegen der Workbench-Einstellungen war damals jedoch noch in einem Stück vorhanden, es gab lediglich verschiedene Symbole, damit man in verschiedene Fenster des Programmes einsteigen konnte. Seit der Workbench 2.0 ist aber auch das Preferences-Programm selbst in seine Einzelzeile aufgeteilt worden und in diesem Verzeichnis abgelegt. Mit der Version 2.0 fanden zwei weitere Directories Platz auf der Workbench: "WBStartup" und "Monitors". Normalerweise ist auf der Workbench noch ein weiteres Symbol zu finden, das der Shell. Sollten Sie nicht fündig werden, so schauen Sie doch einmal im Verzeichnis "System" nach. Doch zu diesem Symbol und was sich dahinter alles verbirgt, erfahren Sie im nächsten Kapitel mehr.

Wenn auch wegen der größeren Datenmenge verschiedene Dateien und Verzeichnisse von der eigentlichen Workbench-Diskette ausgelagert werden mußten, gehören diese im Grunde dennoch zur Workbench. In diesem Kapitel werden daher auch alle anderen Systemdisketten bzw. deren Inhalt angesprochen.

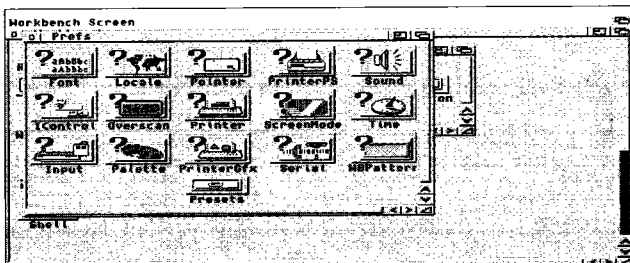
Widmen wir uns zunächst einmal den einzelnen Verzeichnissen und deren Inhalte.

2.2.1 - Die Voreinstellungs-Programme

Leider (oder besser: zum Glück) haben sich die Voreinstellungsprogramme der einzelnen Workbench-Versionen so stark geändert, daß wir hier nicht mehr auf die Preferences der Versionen 1.2 und 1.3 eingehen können, zu groß sind inzwischen die Unterschiede. Es existiert jedoch hierfür auch umfangreiche Literatur (wie etwa das Buch "Amiga-Der Einstieg", das in der Media Verlagsgesellschaft erschienen ist).

2.2.1.1 - Einstellungen unter Version 2.0 - Unterschiede bei Version 3.0

Seit der Workbench 2.0 sind die Preferences-Programme wirklich einzelne kleine Programme, wobei für jeden Bereich ein eigenes Preferences-Programm vorgesehen ist. Es werden auch normalerweise bei neuen Hardwareerweiterungen Voreinstellungs-Programme mitgeliefert, die Sie auch in diese Schublade kopieren sollten.



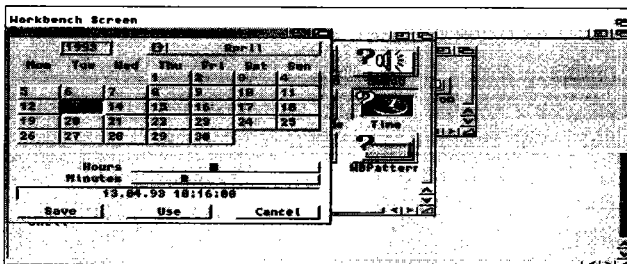
Ab der Workbench 2.0 sind die Voreinsteller echte kleine Programme

Wenn Sie jedoch stolzer Besitzer der Workbench 3.0 sind, werden Sie vergeblich im Prefs-Ordner der Workbench-Diskette nach diesen Programmen Ausschau halten, da hier auch für diese kein Platz mehr vorhanden war. Sie wurden kurzer Hand auf die ExtrasD-Diskette ausgelagert. Die Einstellung der Workbench fordert deswegen einige Geduld. Sie können entweder versuchen, eine Workbench-Diskette zu erstellen, auf der alle überflüssigen

Programme gelöscht werden, um Platz für die Voreinsteller und andere wichtige Daten wie Druckertreiber oder Monitore zu schaffen. Oder Sie nehmen unzählige Diskettenwechsel in Kauf und arbeiten mit dem Prefs-Ordner der ExtrasD. Zur Einführung ist sicherlich der zweite Weg günstiger - wenn auch arbeitsintensiver - doch können Sie diesen Weg ohne Vorkenntnisse ohne weiteres beschreiben. Auf die Dauer werden Sie dann wohl oder übel eine eigene Workbench-Diskette erstellen, die dann auch diesen Programmen Platz bietet. Bei der Besprechung der Preferences werden wir gleich einige Neuerungen in der Bedienung von AmigaOS 2.0 allgemein kennenlernen. Bitte bedenken Sie, daß sämtliche Text-, Gadget- und Menübezeichnungen im Folgenden mit ihrer englischen Originalbezeichnung genannt sind. Sollten Sie bereits über die Workbench 2.1 oder 3.0 verfügen und eine andere Sprache ausgewählt haben, unterscheiden sich die Anzeigen natürlich. Doch jetzt ist es an der Zeit, die einzelnen Programme zu betrachten.

2.2.1.2 - Time

Auch wenn Sie eine Echtzeituhr besitzen, so muß diese zumindest ein erstes Mal eingestellt werden. Dazu (und zur Einstellung nach jedem Neustart, wenn Sie keine besitzen) dient der Voreinsteller "Time". Er eignet sich für den Anfang besonders gut, weil wir ganz genau wissen, was wir nun einstellen wollen - eben die aktuelle Uhrzeit - und er bereits Neuerungen enthält, die erläuterungsbedürftig sind. Starten Sie also das Programm "Time", das sich im Ordner "Prefs" befindet.



Mit Time stellen Sie die Systemuhrzeit ein.

Gleich oben ist in einem Kästchen ein Monatsname zu lesen und links davon ein gebogener Pfeil, der auf sich selbst zeigt. Dieses neues Symbol zeigt einen

Gadget-Typ an, der jedesmal, wenn er angeklickt wird, seinen Inhalt ändert. Dadurch kann man sich im Fall der Monatsnamen durch wiederholtes Klicken bis zum Monat vorarbeiten, den man möchte. Ist man bei Dezember angelangt, wird automatisch wieder von vorne - in diesem Fall eben Januar - begonnen. Diese neuen Gadgets, die im übrigen "Blättersymbole" genannt werden, werden Sie in Zukunft noch öfter antreffen. Oftmals ist es durchaus sinnvoll, erst einmal alle Möglichkeiten durchzuklicken, damit man überhaupt weiß, was alles zur Auswahl steht. Stellen Sie nun also den richtigen Monatsnamen ein. Dabei verändert sich die darunterliegende Anzeige des Kalenders, um die korrekte Anzahl von Tagen und die Verteilung der Wochentage wiederzugeben. Klicken Sie einfach auf den aktuellen Tag, Neues gibt es hier nicht zu berichten. Ebenso verhält es sich mit der Jahreszahl, die ganz einfach ein Textgadget darstellt, in dem Sie mittels Tastatur das Jahr eingeben können. Schließlich sind Schieberegler zur Einstellung der Uhrzeit vorhanden. Mit den drei unteren Gadgets können Sie nun das Ergebnis Ihrer Bemühungen entweder in die Echtzeituhr übernehmen ("Save"), die Systemuhr setzen, aber nicht die Echtzeituhr verändern ("Use") oder ganz verwerfen ("Cancel"). Diese drei Gadgets sind auch in allen anderen Preferences-Editoren vorhanden und bedeuten allgemein immer Abspeichern, Benutzen ohne abzuspeichern oder Verwerfen. Bei den anderen Programmen werden jedoch die zu sichernden Daten auf die Diskette geschrieben und haben nichts mehr mit der Echtzeituhr zu tun. Doch verlassen wir nun den Time-Editor und wenden uns dem nächsten Preferences-Utility zu.

2.2.1.3 - Input

Wesentlich interessanter wird es beim Input-Editor. Die Einstellungen der "Mouse-Speed", die "Double-Click"-Zeit sowie die "Key Repeat"-Einstellungen sind von 1.3 übernommen worden.

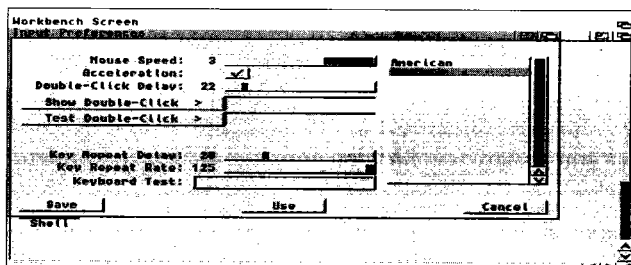
Um sie kurz zusammenzufassen: Mit "Mouse-Speed" läßt sich die Sensibilität der Mausbewegungen definieren: von langsam (bei Workbench 2.0 ist dies "4", bei Version 3.0 ist es "1") bis schnell ("1" bei 2.0 und "3" bei Version 3.0).

Die "Double-Click"-Zeit ist der Zeitraum, innerhalb dessen ein "Double-Click" noch erkannt wird. Hiermit können Sie also die Zeit festlegen, innerhalb der Sie beispielsweise ein Icon auf der Workbench zweimal anklicken

müssen, um das ein Programm zu laden. Hier empfiehlt sich ein Wert aus der goldenen Mitte, denn wenn Sie sich zu wenig Zeit vorgeben, werden Sie wohl nur unter großen Mühen ein Programm starten können. Ist andererseits das Zeitintervall zu großzügig bemessen, kann es vorkommen, daß Sie ein Programm starten, daß Sie eigentlich gar nicht wollten (weil Sie beispielsweise nur nachprüfen wollten, ob das Icon bereits einmal angeklickt wurde oder nicht).

“Key Repeat Delay” gibt an, nach welcher Zeit die Tastenwiederholung eingeschaltet wird, die dann die Funktion der Taste mit der Geschwindigkeit, die durch “Key Repeat Speed” eingestellt ist, wiederholt, bis die Taste wieder losgelassen wird. Sehr hilfreich ist dabei die neue Möglichkeit, seine Einstellungen gleich auszuprobieren. Dazu gibt es das “Show”-Gadget, nach dessen Betätigung das rechts davon liegende Feld seine Farbe für genau den Zeitraum ändert, wie Sie in “Double-Click” eingestellt haben, was weit anschaulicher ist, als ein trockener Zahlenwert. Sie können versuchen, das Feld “Test” mit einem Doppelklick anzuwählen. Gelingt es Ihnen, wird nebenan “Yes” oder “Double-Clicked” gemeldet, andernfalls erscheint “No” oder “Too slow”. Auf diese Weise können Sie überprüfen, ob die gewählte Einstellung für Sie geeignet ist.

Ebenso hat man in dem mit “Key Repeat Test” - es wird bei der Workbench 3.0 “Keyboard Test” genannt - benannten Text-Gadget Gelegenheit, einen Text einzutippen und dabei die Tastenwiederholungsfunktion auszutesten.



Die Parameter für die Maus und Tastatur werden mit “Input” festgelegt.

Der Input-Editor hat eine weitere sehr interessante neue Möglichkeit zu bieten: die Mausbeschleunigung (“Acceleration”). Einmal ist der verwendete Gadget-Typ an sich neu, denn wenn Sie ihn anklicken, erscheint ein kleiner Haken darin, der anzeigt, daß die entsprechende Funktion angewählt wurde.

Und auch die Funktion selbst ist bemerkenswert: die "Acceleration" bewirkt nämlich, daß der Mauszeiger sich bei kleinen Auslenkungen nur langsam bewegt, fährt man jedoch weitere Strecken, so wird er zunehmend schneller. Dies ist eine äußerst sinnvolle Neuerung, die bisher nur mittels zusätzlicher Hilfsprogramme aus dem PD-Bereich installiert werden konnte und eine echte Bereicherung darstellt. Denn es ist damit möglich, sehr genaue Bewegungen auszuführen und dennoch schnell von einem Ende des Bildschirms zum anderen zu kommen. Ganz gleich, welche Mausgeschwindigkeit Sie bevorzugen, ist dies also eine empfehlenswerte Funktion, die Sie mit Sicherheit bald nicht mehr missen möchten. Im Input-Editor der Workbench-Versionen 2.1 und 3.0 ist zusätzlich noch ein Auswahlfeld zur Festlegung der Tastaturbelegung vorhanden. Ab der Version 2.1 existiert das Programm "Setmap" nicht mehr, das vorher in der Startup-Sequence dafür gesorgt hat, daß nach dem Starten des Amigas die gewünschte Tastaturbelegung vorhanden war. Die Tastaturbelegung wird nun wie alle anderen Preferences behandelt. Die Bedienung dürfte keine Schwierigkeiten bereiten: Mit dem Rollbalken können Sie versteckte Tastaturbelegungen in den sichtbaren Bereich schieben und die gewünschte einfach anklicken, worauf sie bei der Version 2.1 in dem darunterliegenden Kästchen erscheint. Bei der Version 3.0 erkennt man die aktuelle Tastaturbelegung daran, daß sie farblich in der Liste hervorgehoben ist. Sonst bleibt alles beim alten.

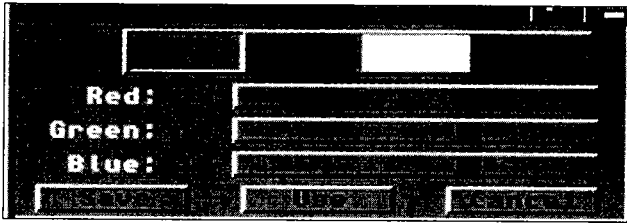
2.2.1.4 - Palette

Dieser Farb-Editor hat die größte Veränderung zwischen der Version 2.0 und 3.0 erfahren - zum guten, wie Sie sich gleich überzeugen können.

Bei der Version 2.0 ist die Einstellung der Workbench-Farben nicht atemberaubend. Am oberen Ende können Sie eine der vier Farben wählen, die Sie zu verändern beabsichtigen und dann mit den drei Schiebereglern unterhalb die Rot-, Grün- und Blauwerte verändern.

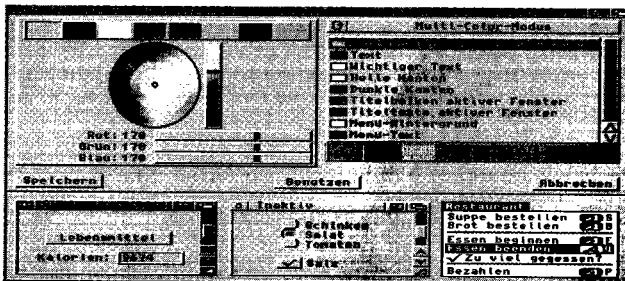
Viel läßt sich dazu nicht sagen, am Anfang muß man mit den Reglern wohl ein wenig spielen, bevor man das richtige Gefühl bekommt, um die Farbe, die man sich vorgestellt hat, auch auf den Bildschirm zu zaubern. Hilfreich er-

weist sich dabei das "Edit-Menü", dessen Untermenü "Presets" einige Vorschläge enthält, die Sie einmal durchprobieren können. Wenn Ihnen dann eine Farbpalette besonders gut zusagt, können Sie diese ja übernehmen, gegebenenfalls noch leicht modifizieren und dann abspeichern bzw. verwenden.



Bei der Workbench 2.0 und 2.1 versehen noch die alten Schieberegler ihren Dienst

Bei dem Palette-Editor der Workbench 3.0 hat man sich nun bemüht, die Farbeinstellungen wesentlich einfacher zu gestalten, hat der Benutzer ja auch wesentlich mehr Farben zur Verfügung.



Fast nicht mehr zu erkennen: der Farbeinsteller der Workbench 3.0 mit dem auffallenden Farbrad.

Zu Beginn fällt gleich der farbenprächtige Kreis in der linken Hälfte des Bildschirms auf, darin befindet sich ein kleiner schwarzer Ring, der sich in etwa an der Stelle befindet, die ungefähr den Farbton der aktuellen Farbe aufweist. Wenn Sie nun mit der Maus irgendwo innerhalb dieses Kreises einen Punkt anklicken, so springt zum einen der schwarze Ring an diese Stelle, zum anderen ändern sich die Balken-Gadgets (der englische Fachbegriff dafür lautet "Proportional-Gadget") unterhalb und rechts neben dem Kreis. Gleichzeitig wird auch die Farbe des eingerahmten Kästchens über dem Kreis

verändert. Mit dieser Kästchenreihe legen Sie fest, für welche Farbe Sie die Einstellung vornehmen wollen. Für Benutzer, die von älteren Versionen die Farbeinstellung über die jeweiligen Rot-, Grün- und Blau-Anteile gewöhnt sind, existieren nach wie vor die drei Balken-Gadgets unter dem Farbkreis. Im Gegensatz zu den älteren Versionen ist jedoch die Abstufung dieser Gadgets feiner gestaltet worden, ein Gadget besitzt nun 256 Schritte, während die älteren Versionen lediglich 16 Schritte angeboten hatten.

Des weiteren ist das senkrechte Proportional-Gadget eine Neuheit, denn mit diesem Gadget kann die Helligkeit des Farbtons angegeben werden. Der Amiga rechnet dann automatisch die Einstellung, die mittels des Farbkreises vorgenommen wurde und die Helligkeit um in die jeweiligen Rot-, Grün- und Blau-Anteile - die waagrechten Balken werden entsprechend verändert.

Diesen drei Balken kann jedoch auch eine andere Bedeutung zugewiesen werden. Wählen Sie dazu im Menü-Titel "Settings" den Untermenüpunkt "HSB" des Menüpunktes "Slider Color Model" an. Wenn Sie mit einem deutschsprachigen Menü arbeiten, ist es entsprechend des Untermenüpunkt "FSH" des Menüpunktes "Farb-Schieberegler" des Titels "Optionen". Nun werden die Werte, die links neben den jeweiligen Gadgets erscheinen nicht mehr als Farbanteile interpretiert, sondern sie geben den Farbton, Stättigung und Helligkeit an. Auch so können Sie alle möglichen Farben erstellen, es bedarf nur noch etwas mehr Übung.

Eine weitere hervorragende Neuheit ist die Möglichkeit, die Wirkung der neuen Farbe in Beispielen zu betrachten. Wenn die Größe der Workbench nicht ausreichen sollte, werden nach dem Anklicken des Gadgets "Show samples" Beispiele für Fenster, Gadgets und Menüs angezeigt. Wenn jedoch genügend Platz ist, so wird das Gadget von vorn herein weggelassen und stattdessen die Beispiele von Beginn an gezeigt.

Bei der Version 3.0 gibt es eigentlich nichts mehr, was nicht eingestellt werden kann. So hat der Benutzer hier die Möglichkeit, neben den Farben für den Text in der Workbench auch die in den Titelleisten von aktiven und deaktivierten Fenstern, von Menüs und vielem mehr festzulegen. Dazu klickt er zunächst eines der Kästchen in der rechten Hälfte des Bildschirmes an und wählt anschließend unterhalb die Farbe an, die er dafür verwenden möchte. Der individuellen Gestaltung der Workbench sind hier wirklich keine Grenzen mehr gesetzt.

Bemerkenswert ist eine weitere Eigenschaft der Preferences-Programme der neuen Generation, die es ermöglichen, bestimmte Voreinstellungen abzuspeichern, ohne daß diese Festlegungen sofort bei jedem Neustart aktiviert werden. Es kann vorkommen, daß für bestimmte Anwendungen einige Farbzustammenstellungen besser geeignet sind als andere und bei anderen Programme schließlich wieder andere Einstellungen vorteilhafter sind.

Sie haben nun die Möglichkeit, diese Farbzusammenstellungen so zu speichern, daß ein Doppelklick auf das entsprechende Gadget die Farben sofort anpaßt, ohne daß Sie lange wieder mit den Voreinstellungen herumexperimentieren müssen. Zu diesem Zweck besitzt jedes Programm im Menü "Project" den Punkt "Save AS". Hier geben Sie den Namen des Gadgets - und damit den der Einstellung - an, unter dem die Einstellungen abgespeichert werden sollen. Wenn Sie nun im Menü "Settings" den Punkt "Creat Icons" (bei den Programmen der Version 2.0 heißt dieser Punkt "Save Icons") aktiviert haben, wird die Datei mit einem Gadget versehen. Sie sind dann in der Lage, diese Einstellung durch einen einzigen Doppelklick zu aktivieren.

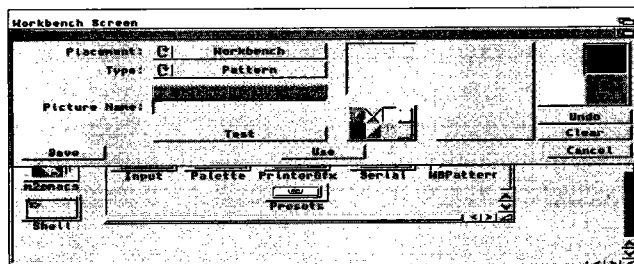
PAG Sandini

2.2.1.5 - WBPatten

Seit der Version 2.0 hat der Benutzer auch die Möglichkeit, den Hintergrund mit Mustern ("Pattern") zu belegen, die man mit diesem Editor festlegen kann.

Es ist möglich, unterschiedliche Muster für das Workbench-Fenster sowie die anderen Fenster zu wählen. Ab der Version 3.0 kann sogar der Screen mit einem eigenen Muster belegt werden. Bei geschickter Wahl ist es so möglich, das WB-Fenster leichter von anderen Fenstern zu unterscheiden. Je nachdem, für welchen Bereich Sie nun ein Muster wählen wollen, klicken Sie bei der Version 2.0 einen der beiden Knöpfe links oben an, die entweder mit "Workbench" oder mit "Windows" bezeichnet sind.

Diese Knöpfe, die immer vertikal zueinander angeordnet sind, können als ein Gadget betrachtet werden, da sie in der Regel gegenseitig ausschließen, so daß das Anwählen der einen Option die anderen deaktiviert. Sie sind ebenfalls mit der Version 2.0 neu eingeführt worden.



WBPattern macht Schluß mit langweiligen Fenstern und Bildschirmen.

Das Programm der Version 3.0 weist ein Blätter-Gadget auf, um auszuwählen, welches Muster Sie editieren wollen. Als dritte Möglichkeit kann hier auch für den Bildschirm selbst ein eigenes Muster festgelegt werden, weshalb das Blätter-Gadget noch das Wort "Screen" in der Liste hat.

Doch nun zur Handhabung des Pattern-Editors selbst. Sie sehen zwei verschieden große Kästen, der kleinere gibt das Muster in Original-Größe an, der große beinhaltet eine Vergrößerung davon. Da die Vergrößerung ein genaueres Arbeiten ermöglicht, können Sie nur in diesem Kasten die Einstellung verändern. Dazu wählen Sie eine Farbe aus der davon rechts liegenden Farbpalette, die dann bei der Workbench 2.0 im darüberliegenden Kasten gezeigt wird, bei der Version 3.0 wird die Farbe eingerahmt.

Anschließend klicken Sie einfach in das Editorfeld hinein, um einen Punkt in der gewählten Farbe zu setzen. Halten Sie die Maus gedrückt, so können Sie freihändig zeichnen. Der jeweils letzte Linienzug wird dabei zwischengespeichert, so daß Sie, falls Ihnen ein Fehler unterlaufen ist, diesen mit "Undo" rückgängig machen können.

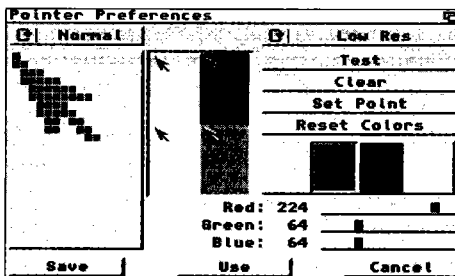
Einen Eindruck, wie das Ganze dann am Ende aussehen wird, vermittelt Ihnen die kleine Anzeige. Dort sind zwölf solcher Einheiten zusammengefügt, was auch hilft, Ränder korrekt zu setzen, damit ein nahtloser Übergang garantiert ist. Für die weniger Kreativen sind wieder einige Vorgaben vorhanden, die Sie durch einfaches Anklicken in den Editor übernehmen können.

Damit Sie nicht jedesmal den Editor mit "Use" verlassen müssen, um ein ganzes Window in seiner neuen Pracht zu sehen, ist es möglich, das gewählte Muster mittels "Test" in das entsprechende Fenster bzw. bei Version 3.0 ent-

sprechend auch in den Screen zu übernehmen. Wenn Ihr Entwurf vollends daneben ging, kann mit dem Clear-Gadget das Editorfeld schließlich noch komplett gelöscht werden, um nochmals ganz von vorne beginnen zu können. Das Pattern-Programm der Workbench 3.0 bietet noch eine weitere Möglichkeit. Hier kann man nämlich auch ein beliebiges Bild auf ein Fenster oder den Screen legen. Hierzu wählen Sie in dem unteren der beiden Blättergadgets das Wort "Picture" aus. Darunter erscheint das Gadget "Select Picture" anschließend in normaler Schrift, nach einem Klick auf das Gadget erscheint ein sogenannter File-Requester, in dem Sie dann angeben können, welches Bild Sie verwenden möchten. Wenn Sie nun nach der Auswahl des Bildes das Gadget "OK" anklicken, so erscheint der Dateiname im Feld "Picture Name". Sie können dann genau so weiter machen, wie bei einem selbsterstellten Muster.

2.2.1.6 - Pointer

Hier lohnt es sich, wieder die Prefereces-Programme der beiden Versionen getrennt zu betrachten. Wie auch bei Kick 1.3 war der Pointer-Editor bereits auf einem eigenen Bildschirm ausgelagert, der dem unter 2.0 erscheinenden auffallend gleicht. Für die Aufsteiger gibt es hier also nichts neues, für alle anderen ein paar kurze Erklärungen: Das Editieren, also Zeichnen an sich entspricht genau dem des Pattern-Editors. Darüber hinaus können Sie noch die Farbpalette des Mauszeigers ändern, was genauso wie das Ändern im Palette-Editor vonstatten geht. "Clear" löscht das Editorfeld und "Reset Color" stellt die gespeicherten Farben wieder her. Beachtung verdient nur die "Set Point"-Funktion, mit der Sie den Auslösepunkt des Mauszeigers festlegen.



Wie bei der guten alten Workbench 1.2 - der Mauszeiger-Editor

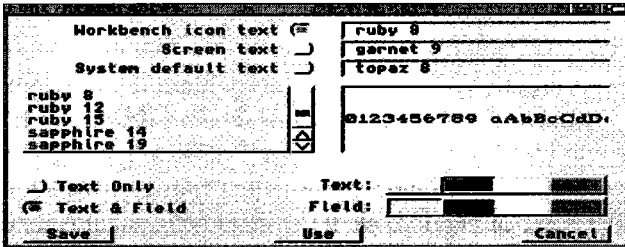
Bei der Workbench 3.0 nun fallen gleich zwei neue Blätter-Gadgets ins Auge. Hier kann nicht nur der normale Mauszeiger verändert werden, sondern auch das Warte-Symbol (hierfür ist eine Uhr voreingestellt). Das zweite Gadget ist nicht minder interessant. Dieses Blätter-Gadget enthält die Begriffe "Low Res" und "High Res" in seiner Liste. Dabei sind die Namen nicht ganz korrekt, den eigentlich würde der langjährige Amiga-Freund erwarten, daß der Pfeil zwei verschiedene Auflösungen bewältigen kann. In Wirklichkeit gibt es jedoch zwei verschiedene Größen des Zeigers. Bei Wahl des "Low Res"-Zeigers, erscheint der gewohnte Pfeil im Editorfeld, bei "High Res" jedoch wird jeweils die mögliche Breite und Höhe des Pfeiles verdoppelt, der Zeiger kann damit also viermal größer dargestellt werden als bisher. Für alle anderen Funktionen gelten die obigen Beschreibungen analog.

2.2.1.7 - Font

Mit dem Betriebssystem 2.0 wurde es erstmals möglich, andere Zeichensätze als den Standard-Font "Topaz" in der Workbench-Umgebung zu benützen. Für den Anwender von Disketten entsteht dabei aber das Problem, daß die entsprechenden Zeichensätze nicht mehr auf der Workbenchdiskette Platz gefunden haben, sondern auf eine extra Font-Diskette ausgelagert wurden. Damit sind sie nicht von vornherein zugänglich, sondern müssen erst auf die Workbench-Diskette kopiert werden. Dies wiederum geht aber nur von der Shell aus (hier jedoch sehr schnell) oder aber Sie verwenden die Möglichkeit der Workbench, durch Auswahl des Menüpunktes "Show - all Files" auch die Dateien und Verzeichnisse ohne Symbole auf der Workbench darstellen zu können. Dabei müssen die Daten unbedingt in ein Verzeichnis mit dem Namen "fonts" im Hauptverzeichnis der Workbench kopiert werden.

Dies wiederum wirft ein weiteres Problem auf, da die Diskette bereits voll ist. Für den Festplattenbesitzer bestehen diese Einschränkungen nicht, da ja die Daten aller Disketten bereits auf seine Platte kopiert wurden.

Es können verschiedene Fonts für Icon-, Screen- und System-Texte gewählt werden. Dazu wählen Sie das entsprechende Gadget aus und wählen anschließend den gewählten Zeichensatz an - da im Normalfall mehr Zeichensätze zur Verfügung stehen, als im Feld dargestellt werden können, sollten Sie doch einmal den Rollbalken betätigen. Eine Kostprobe der ausgewählten Schriftart wird in einem Kasten dargestellt.



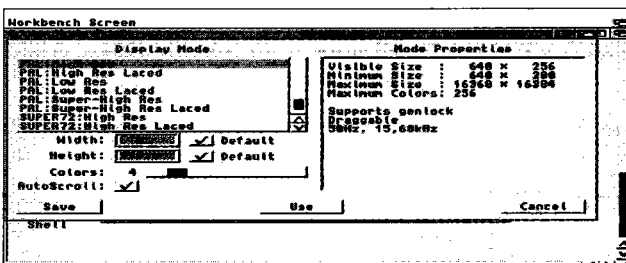
Auch bei den Workbench-Fonts können Sie ein Wörtchen mitreden.

Sie können weiterhin noch die Text- und Hintergrundfarbe festlegen. Dazu sind die "Text" und "Field" Farbpaletten zuständig. Bei Wahl einer Hintergrundfarbe mit dem "Text & Field"-Knopf werden die Icon-Texte mit dieser Farbe unterlegt, wobei ein etwaiges Muster übermalt wird. Sollten Sie also einen Einbettung der Texte in den Hintergrund bevorzugen, ist es besser "Text Only" zu wählen, als das Field-Feld auf die normale Hintergrundfarbe zu setzen, da auf diese Weise das Muster erhalten bleibt.

PAG Sandini

2.2.1.8 - Screenmode

Auch wenn durch das neue AAA-Chipset beim Amiga 1200 weitere neue Bildschirm-Auflösungen verfügbar wurden, so hat sich doch das "Screenmode"-Programm nicht wesentlich verändert, lediglich wurde die Anordnung der Darstellung etwas überarbeitet und die Bezeichnung einiger Gadgets etwas modifiziert.



Die neuen AAA-Chips bieten viele neue Auflösungen.

Es werden die möglichen Bildschirmmodi für Ihre Hardware angezeigt und Sie können die bevorzugte wählen. Im wesentlichen hängt das Angebot davon ab, ob Sie beim OS 2.0 das sogenannte ECS installiert haben - beim OS 3.0 wird das Angebot von den im Verzeichnis Devs:Monitors vorhandenen Monitor-Treibern bestimmt. Sie müssen dabei außerdem unbedingt beachten, daß für einige Modi ein besonderer Monitor benötigt wird, da der übliche 1081 oder 1084 von Commodore nicht flexibel genug ist, um alle vom Amiga kommenden Signale verarbeiten zu können. Mittlerweile bietet Commodore zwei weitere Monitore an, den A2024 und A 1960.

Die Aufzählung der verfügbaren Modi ist hier eigentlich überflüssig, da Sie sie der Liste des Screenmode-Programm entnehmen können. Sie können die Namen der Bildschirmmodi anklicken. Im Feld "Mode Properties" (oder "Properties of the Selected Mode" bei 2.0) rechts daneben wird beschrieben, welche Anforderungen gestellt werden, ferner ist die Bildaufbaufrequenz und die Unterstützung eines Genlocks angegeben. Desweiteren finden Sie die Anzahl der dargestellten Bildpunkte, die maximale Anzahl von Farben und den größt- bzw. kleinstmöglichen Screen, der bei dem gewählten Modus möglich ist.

Da einige Monitore sowohl PAL- als auch NTSC-Auflösungen verarbeiten können, können Sie beide Modi verwenden. Sie müssen lediglich den entsprechenden Monitor-Treiber in den "Monitors"-Ordner legen. Vorteil hoher Auflösungen ist natürlich, daß mehr Informationen dargestellt werden können. Allerdings unterstützen einige Programme die neuen Modi noch nicht. Und auch der NTSC-Modus hat seinen Vorteil. Sie verlieren zwar 56 bzw. 112 Zeilen, doch dafür gewinnen Sie 10 Hz Bildwiederholfrequenz. Der PAL-Modus läuft nämlich mit 50 Hz und der NTSC-Modus mit 60 Hz. Und je höher die Bildwiederholfrequenz ist, desto ruhiger wird das Bild.

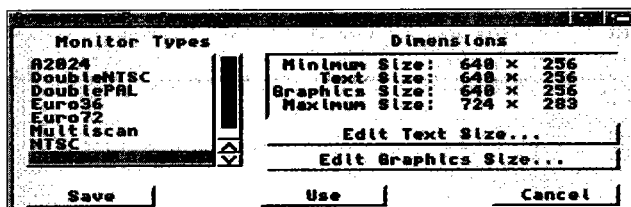
Sollten Sie also vorhaben, längere Zeit vor dem Rechner zu sitzen und vielleicht Texte zu lesen oder zu schreiben, so tun Sie Ihren Augen mit dem NTSC-Modus einen echten Gefallen. Allerdings funktioniert dieser erst mit dem FAT-Agnus Chip, und nicht mit dem ganz alten nur 512 KByte Chip-RAM unterstützenden Agnus 8372. Besitzer des Amiga 1200 brachen sich aber darüber keine Gedanken mehr zu machen, diese besitzen garantiert den neuen Chip.

Interessant ist die Möglichkeit, die Screen-Größe selbst festlegen zu können. Grundsätzlich werden als Default-Werte genau die gewählt, die auch die

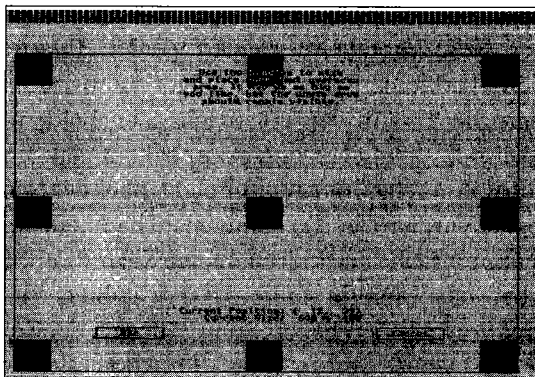
Auflösung angeben, also beispielsweise 640 x 256 Pixel in PAL:HighRes. Doch nun können diese Vorgaben verändert werden. Dadurch läßt sich der nutzbare Bereich verkleinern (was weniger sinnvoll ist) aber auch vergrößern. Wenn Sie also in den entsprechenden Feldern beispielsweise 2000 x 1000 Pixel oder ähnliches eingeben, bekommen Sie einen riesigen Bildschirm - natürlich wächst Ihr Monitor nicht, aber der Screen, auf dem ja die Windows liegen, wird größer. Da er nun nicht mehr auf den Monitor paßt, können Sie ihn ganz einfach verschieben, indem Sie mit der Maus in die Nähe des Randbereiches des Monitor fahren. Dazu sollten Sie allerdings die "Autoscroll"-Funktion aktiviert haben, sonst läßt sich der Screen nur mittels Menüleiste oder der "Mouse Screen Drag"- Funktion (siehe dazu auch unter IControl) verschieben. Auch wenn nur die wenigsten von dieser Anwendung Gebrauch machen werden, sollten Sie es sich zumindest einmal ansehen: Es ist ein atemberaubendes Erlebnis (vor allem unter einem schnelleren Prozessor wie etwa einem 68030). Zu guter Letzt kann auch die Anzahl der darstellbaren Farben verändert werden, da aber mehr als vier Farben kaum unterstützt werden und Sie zudem Geschwindigkeitseinbußen und vergrößerten Speicherbedarf hinnehmen müßten, ist vom Gebrauch dieser Funktion abzuraten.

2.2.1.9 - Overscan

Unter "Overscan" versteht man das Aufziehen des Bildschirms bis in die äußersten Ecken des Monitors. Daß dieser Bereich grundsätzlich nicht genützt wird, hat aber auch seinen guten Grund: Wenn Sie nicht gerade an einem Flatscreen-Monitor sitzen, wird das Bild in den Ecken doch schon arg verzerrt, zudem kann es sein, daß gerade bei einem angeschlossenen Fernseher die Randbereiche gar nicht mehr dargestellt werden.



Damit Sie immer richtig im Bild sind...



Passen Sie mit Overscan Ihren Bildschirm dem Monitor an.

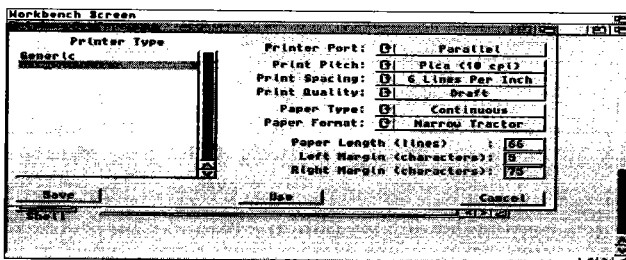
Wenn Sie aber auch das letzte Eck ausnützen möchten, so bietet sich dieser Voreinsteller an. Dazu wählen Sie zunächst den Bildschirmmodus, für den Sie die Einstellung machen möchten (PAL, NTSC, ...). Anschließend können Sie entweder den "Text Overscan" oder den "Standard Overscan" editieren. In beiden Fällen wird ein Bildschirm angezeigt, auf dem neun Quadrate verteilt sind.

Mit den vier an den Seiten können Sie die Höhe und Breite verändern, indem Sie sie mit der Maus ziehen. Mit den in den Ecken lassen sich jeweils zwei aneinanderstoßende Ränder gleichzeitig verschieben. Und in der Mitte kann der gesamte Screen verschoben werden. Vor allem für Besitzer eines Multiscan Monitors ergibt sich eine interessante Variante: Wählen Sie aus den "Screenmodes" den NTSC-Modus und vergrößern Sie mittels "Overscan" die Höhe des Bildschirms auf 232 Zeilen. Da beispielsweise der NEC 3D den Bildschirm in NTSC nicht auf volle Höhe auszieht, können Sie immer noch alle Bereiche des Screens sehen: alle 232 Zeilen. Dabei haben Sie nun eine Bildwiederholfrequenz von 60 Hz, ein guter Tausch für 24 verlorene Zeilen! Es gibt noch einige andere Monitore, die diese Einstellung verarbeiten können - probieren Sie es einfach aus, ob es auch bei Ihnen klappt.

2.2.1.10 - Printer

Mit dem Printer-Editor kommen wir nun das erste Mal mit der Außenwelt in Kontakt. Hier läßt sich einer der Drucker auswählen, die Sie bei der Installation der Workbench 2.0 angegeben haben. Bei Verwendung der Workbench 3.0 müssen Sie den gewünschten Druckertreiber erst vom Verzeichnis "Printers" der "Storage"-Diskette in das Directory "Devs/Printers" kopieren. Wie auch bei den Fonts gilt hier für die Benutzer, die den Rechner mit einer Diskette starten müssen, daß im Normalfall erst einmal Platz geschaffen werden muß, bevor der Druckertreiber kopiert werden kann. Wenn die Treiber erst einmal zur Verfügung stehen, dann haben Sie die freie Auswahl.

Zusätzlich zur Einstellung des richtigen Druckers können anschließend noch Angaben zur Länge des verwendeten Papiers in Zeilen ("Paper Length (Lines)"; bei Endlospapier meist 72, bei DIN A 4 66) und dem linken und rechten Rand ("Left Margin (Chars)" und "Right Margin (Chars)") gemacht werden. Auch grundsätzliche Einstellungen, wie etwa den "Printer Port", der angibt, ob die Daten an die serielle oder die parallele Schnittstelle geschickt werden sollen, oder ob Sie Endlospapier bzw. Einzelblatt verwenden ("Paper type") und welche Breite das Papier hat ("Paper Size").

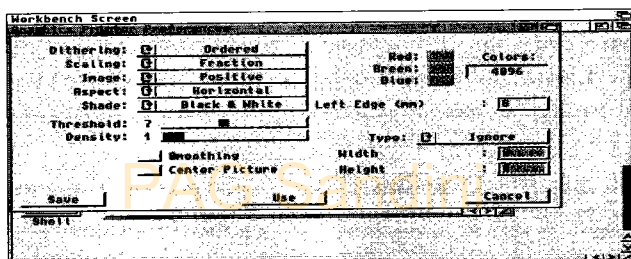


Damit Ihr Drucker alles korrekt ausspuckt, sollten Sie den Amiga richtig einstellen - mit dem Printer-Preferences

Von Anwendung zu Anwendung hingegen können die restlichen Parameter noch abweichen, nämlich die zu verwendende Zeichendichte ("Printer Pitch"), die von normal ("10-Pica") über schmal ("12-Elite") bis hin zu sehr schmal ("15-Fine") reichen kann. Für Briefe werden Sie sicherlich normal oder bestenfalls noch schmal verwenden, sollten Sie jedoch Tabellen mit vie-

len Spalten ausdrucken wollen, so können sich die 136 Zeichen pro Zeile, die eine sehr schmale Einstellung auf DIN A 4 bietet, durchaus bewähren. Mit "Print Spacing" ist schließlich noch der Druckabstand, also die Anzahl der Zeilen, die auf einen Zoll (2,54 cm) passen, gemeint und wann Sie bei der "Print Quality" die Schönschrift "Letter" oder Schnellschrift "Draft" bevorzugen, dürfte auf der Hand liegen.

2.2.1.11 - PrinterGfx



Für den Grafikausdruck gibt es einen eigenen Voreinsteller

Da sicher eines Tages der Wunsch nach etwas mehr als nur ödem Text auf dem Papier auftaucht, können Sie mit dem PrinterGfx-Editor die nötigen Konfigurationen zum Grafikausdruck einstellen. Da wäre zunächst einmal die Positionierung der Grafik auf dem Papier. Da in Abhängigkeit von der Größe des auszudruckenden Bildes meistens nicht die volle Breite auf dem Blatt ausgenutzt wird, können Sie unter "Left Offset" (unter 3.0 entsprechend unter "Left Edge") entweder den absoluten Abstand zum linken Rand in 1/10 Zoll, also 2.45 mm angeben, oder "Center Picture" anwählen, wodurch der Ausdruck zentriert wird.

Über die Abmessungen des Ausdrucks können Sie in den Feldern "Width" und "Height" Angaben machen. Diese werden je nach dem Zustand des "Type"-Blättersymbolen interpretiert. "Ignore" ignoriert diese Angaben, "Boundet" legt maximale Grenzwerte fest, "Absolute" erzwingt eine Anpassung an die angegebenen Maße, ebenso "Pixels", nur daß in diesem Fall die Werte in Punkten angegeben sind, nicht in Zoll, und "Multiply"

schließlich sorgt für eine Vergrößerung der Grafik um den angegebenen Faktor. Bei den letzten drei Typen kann der Ausdruck durch explizite Angabe von Höhe und Breite stark verzerrt werden. Deshalb ist es in diesen Fällen eher ratsam, einen der beiden Werte auf Null zu setzen, wodurch eine automatische Anpassung an den anderen Wert vorgenommen wird.

Beachten Sie jedoch, daß bei dem neuen PrintGfx-Editor der Workbench 3.0 die Angaben nicht mehr in Zoll, sondern in Millimeter anzugeben sind bzw. als solche interpretiert werden.

Um zum Beispiel ein 6 Zoll breites Bild zu erzwingen, daß entsprechend seiner Abmessung ausgedruckt und nicht verzerrt werden soll, wählen Sie einfach "Absolute", setzen "Width" auf "6" (bei OS 3.0 entsprechend auf "15"(mm)) und "Height" auf "0" - das wars. In diesem Zusammenhang kann man auch gleich das "Scaling"-Gadget betrachten, das die Option "Fraction" und "Integer" zur Verfügung stellt. Wählt man erstere, so kann das Bild stufenlos vergrößert werden, bei letzterer geht dies nur in Vielfachen der Größe der Grafik, die man ausdrucken will. Diese Einstellung ist immer dann sinnvoll, wenn Bilder mit Linien oder ähnlichem ausgedruckt werden sollen, da so beispielsweise unterschiedliche Strichdicken vermieden werden.

Doch das waren noch lange nicht alle zur Verfügung stehende Optionen. Sollten Sie über einem Farbdrucker verfügen, so könnte es sein, daß die Farben auf dem Ausdruck nicht ganz mit dem Bildschirm übereinstimmen. Um dieses Problem zu korrigieren, gibt es die "Color Correct"-Gadgets. Sie können mit ihnen die Anzahl der Farben aus dem Rot-, Grün- und Blaubereich verringern, was den Ausdruck durchaus verbessern kann. Welche Einstellungen im jeweiligen Fall am besten sind, hängt jedoch sehr stark vom verwendeten Drucker ab und Sie werden wohl einige Probeausdrucke investieren müssen, bis Sie die richtige Einstellung gefunden haben.

Wenn nur ein schwarz-weiß Drucker zur Verfügung steht, so müssen die Farben irgendwie umgerechnet werden. Die Einstellung "Color" im "Shade"-Feld scheidet schon einmal aus. Doch da gibt es ja noch "Grey Scale 1" und "Grey Scale 2" sowie "Black & White". Die letzte Option bewirkt eine Aufteilung aller Farben auf entweder schwarz oder weiß. Wo die Grenzlinie zu ziehen ist, läßt sich mit dem "Threshold"-Regler steuern. Dies ist beispiels-

weise dann notwendig, wenn Sie ein sehr helles oder aber sehr dunkles Bild ausdrucken wollen, da sonst womöglich ein einfarbig weißes oder schwarzes Blatt aus Ihrem Drucker wandert. Sehr viel sinnvoller sind da schon die Graustufenumrechnungen. Da ein Drucker meist eine höhere Auflösung bietet als ein Monitor, kommen auf einen Bildpunkt auf dem Bildschirm mehrere auf dem Ausdruck. Je nachdem wie viele dieser Punkte gesetzt und in welchem Muster sie angeordnet sind, können unterschiedliche Graustufen ausgemacht werden. "Grey Scale 2" ist dabei eher zu vernachlässigen, da es nur vier Farben verarbeiten kann und für Benutzer beispielsweise eines A 2024 Monitors gedacht ist, der ebenfalls nur vier Farben darstellt. Wählen Sie daher am besten "Grey Scale 1".

Sowohl für Farbdrucker als auch für solche, die nur Schwarz-Weiß zu Papier bringen ist das "Dithering" interessant. Da auch Farbdruckern nur eine begrenzte Anzahl von Farben zur Verfügung steht (meist drei oder vier), müssen die restlichen durch Mischen erzeugt werden - und nichts anderes passiert auch bei der Graustufenumrechnung, nur hier werden lediglich die Farben Schwarz und Weiß gemischt. Zur Verfügung stehen die Algorithmen "Orderd", "Halftone" und "Floyd-Steinberg". Gerade die letzte verschluckt eine Menge Rechenzeit, liefert aber meistens auch die anschaulichsten Ergebnisse. Auch in diesem Fall ist wieder einmal Ausprobieren angesagt. Was noch übrig bleibt, sind einige ebenso wichtige wie leicht verständliche Auswahlmöglichkeiten. So wählen Sie mit "Aspect", ob die Grafik um 90 Grad gedreht werden soll, also praktisch vertikal ausgegeben wird, oder aber wie üblich horizontal. Mit "Image" = "Negativ" vertauschen Sie schwarz mit weiß, was gerade bei dunkler Grafik enorm das Farbband schont, bei "Positive" erhalten Sie den Ausdruck mit der Helligkeit, wie das Bild auf dem Monitor erscheint.

Einen anderen Weg, Ihr Farbband zu schonen oder aber die Qualität der Ausdrucke zu erhöhen, stellt der "Density"-Regler dar. Er gibt an, wie "dicht" die Pixel beim Ausdruck angeordnet werden, was zu einer gewaltigen Verbesserung führen kann, jedoch auch den Druck verlangsamt. Auch muß man berücksichtigen, daß nicht jeder Drucker die hohen Einstellungen verarbeiten kann.

Als letzte Möglichkeit, den Ausdruck aufzupeppen, bietet die "Smoothing"-Option an, die bewirkt, daß schräge Linien geglättet werden, wodurch die

Auflösung scheinbar ansteigt. Eine kleine Einschränkung existiert jedoch: Sie können "Smoothing" nicht mit "Floyd-Steinberg" kombinieren. Allerdings bleiben immer noch genügend Kombinationsmöglichkeiten übrig, um Ihnen so viele Probedrucke abzuverlangen, daß von einem Blitzeinstieg keine Rede mehr sein kann. Deshalb an dieser Stelle eine Variante, die sich bewährt hat und die Sie ja immer noch Ihren Wünschen und Ihrem Drucker anpassen können:

Dithering: Halftone; Shade: Grey Scale 1
Scaling: Integer; Smoothing: off
Image: Positive; Center Picture: on
Aspect: Horizontal; Limits: Ignore

Außerdem sollten Sie noch "Density" je nachdem, wie lange Sie warten wollen und was Ihr Drucker so verkraftet, größer als eins wählen.

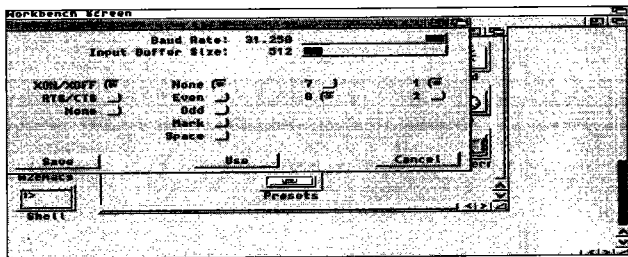
PAG Sandini

2.2.1.12 - Serial

Mit dem Serial-Editor kommen wir wieder zu einer einfacheren Angelegenheit, die aber auch nicht für die Masse der Anwender interessant ist. Es geht dabei um die Konfiguration des seriellen Ports, was in erster Linie wichtig ist, wenn man ein Modem daran angeschlossen hat. Aber auch in diesem Fall werden Einstellungen oftmals vom verwendeten Terminalprogramm vorgenommen, so daß die Grundeinstellungen der Workbench zu meist völlig belanglos sind. Sollte man jedoch den Amiga beispielsweise über Nullmodem mit einem zweiten Rechner verbunden haben, auf dem man dann mittels AUX-Handler eine Remote-Shell eröffnen will, so wird auf eben diese Parameter zurückgegriffen.

Die wichtigste Einstellung ist sicher die Übertragungsrate oder "BAUD-Rate". Sie läßt sich in den üblichen Schritten konfigurieren, wobei aber völlig unverständlich ist, warum der letzte Schritt, die 38400 Baud nicht unterstützt werden. Der Amiga ist mittels des "serial.device" nämlich durchaus zu solchen Raten in der Lage und die unüblichen 31250 Baud werden aus-

schließlich für MIDI und nicht zur Rechnerkopplung genützt. So bleiben für ein Remote-Shell meist nur 19200 Baud übrig, was für diesen Zweck aber auch ausreicht. Sie müssen unbedingt beachten, daß diese und auch alle anderen Einstellungen auf jeden Fall mit denen des angeschlossenen Zweitrechners bzw. der Mailbox übereinstimmen!



Ihre serielle Schnittstelle muß ebenfalls an die angeschlossenen Geräte angepaßt werden.

Lediglich die "Input Buffer Size" können Sie frei wählen, und auch hier gilt: je mehr, desto besser. "Parity", "Bits/Char" und "Stop Bits" sind weitere entscheidende Parameter, die üblicherweise in drei Zeichen abgekürzt werden. So bedeutet "8N1" (was heutzutage am meisten Verbreitung gefunden hat) 8 Bits je Zeichen, "None" und 1 Stopp Bit. Das "Handshaking" hängt vom verwendeten Modem bzw. Kabel ab. So verwenden High-Speed Modems grundsätzlich "CTS/RTS", während ältere Typen noch auf das problematischere "XON/XOFF" zurückgreifen müssen. Bei Verwendung eines Nullmodem-Kabels ist im Grunde überhaupt kein "Handshaking" nötig (also "None"); sollten Sie jedoch eines der besseren Kabel haben, d.h. ein Kabel, bei dem auch die CTS- und RTS-Leitungen belegt sind, so sollten Sie die entsprechende Option wählen.

2.2.1.13 I - Control

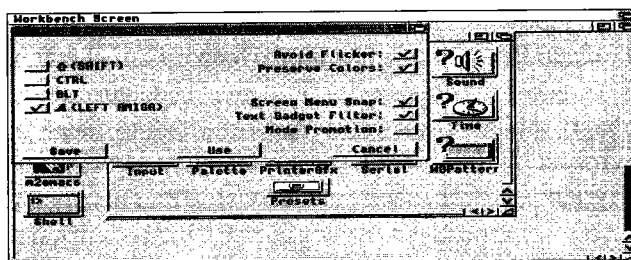
All die Einstellungen, die in kein anderes Preferences-Programm gepaßt haben, finden Sie in "IControl". Dementsprechend weit gestreut sind auch die Themenbereiche. Gleich die erste Kopfleiste ist die sinnloseste überhaupt, weshalb sie bei der Workbench 2.1 und auch bei der neuen Version 3.0 wieder entfernt wurde. Unter dem Titel "Verify Timeout" soll dem Anwender

Gelegenheit gegeben werden, festzulegen, nach welchem Zeitraum das System mit seiner Bearbeitung fortfährt, wenn es auf die Reaktion eines Programms vergeblich wartet. Im Klartext bedeutet dies: Sollte ein Programm derart fehlerhaft programmiert sein, daß es das System warten läßt, bis es schwarz wird, so können Sie einen Stillstand dadurch vermeiden, daß Sie das System zwingen, nach einer gewissen Zeit dennoch weiterzumachen. Daß das natürlich nicht die Lösung aller Probleme sein kann, dürfte klar sein, denn früher oder später wird diese rüde Haltung des Systems Spuren hinterlassen, und zwar bevorzugt solche in roten Kästchen am Kopf des Bildschirms. Wie dem auch sei: Vergessen Sie diese Einstellung! - sofern Sie überhaupt noch die Version 2.0 verwenden.

Etwas sinnvoller werden da schon die Definitionen von Tastaturkürzeln, die jedoch auch nur unter der Version 2.0 veränderbar sind. So können Sie unter "Command Keys" die Tastenkombinationen zum Verlegen des Workbenchscreens und Beantworten von Requestern nach eigenen Vorstellungen gestalten. Da allerdings recht viele Anwenderprogramme ihre eigenen Kürzel für verschiedene Sonderfunktionen haben, und nur die bisher nicht veränderbaren Kombinationen unbelegt ließen, ist das vielleicht auch gar nicht so sinnvoll, denn ob ich nun Amiga-N drücke oder Amiga-" ist doch auch schon egal? Wohl aus diesem Grund kann auch diese Einstellung seit 2.1 wieder nicht mehr vorgenommen werden, weshalb Sie sich jetzt nicht mehr an eigene Abkürzungen gewöhnen sollten.

Anders ist es mit dem "Mouse Drag". Diese Funktion ist endlich wieder einmal sinnvoll. Denn wenn man bisher mit dem Mauszeiger immer auf die Menüleiste eines Screens fahren mußte, um diesen zu verschieben, so geht das jetzt auch einfacher: Sie betätigen einfach die unter "Mouse Drag" festgelegte Taste und dann den linken Mausknopf - und schon können Sie den gesamten Bildschirm verschieben, egal wo sich der Mauszeiger befindet.

Diese Funktion ist auch in Kombination mit einem übergroßen Bildschirm äußerst hilfreich, wenn Sie nicht gerade "AutoScroll" eingeschaltet haben. Denn nur so können Sie in diesem Fall in die entlegene Bereiche des Bildschirms fahren, da die Menüleiste möglicherweise gar nicht sichtbar ist. Wegen des selteneren Gebrauchs der Ctrl-Taste ist diese hierfür recht geeignet, auf jeden Fall vermieden werden sollte es, nur die Shift-Taste alleine einzusetzen, da sonst das Auswählen mehrerer Icons, indem man die Shift-Taste gedrückt hält, logischerweise nicht mehr funktioniert.



Hier legen Sie für verschieden Aktionen die Tastenkombinationen fest.

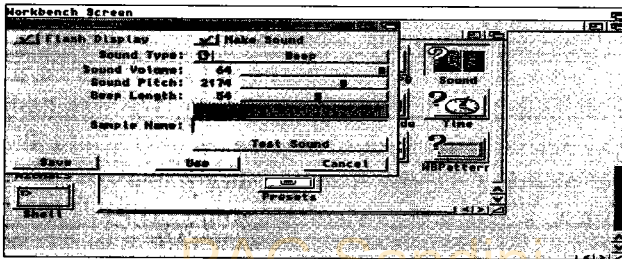
Ebenfalls für Anwender übergroßer Bildschirme ist die Funktion "Screen menu snap" gedacht. Da die Menüs grundsätzlich von links her in die Menüleiste gefüllt werden, kann es bei entsprechend großem Bildschirm schon einmal vorkommen, daß sie im momentanen Ausschnitt nicht erreichbar ist. Wurde jedoch "Screen menu strap" gewählt, so schaltet die Anzeige solange Sie den rechten Mausknopf gedrückt halten in die linke obere Ecke, so daß Sie ohne von Hand nach oben fahren zu müssen, den gewünschten Menüpunkt auswählen können. Lassen Sie den rechten Mausknopf wieder los, springt die Anzeige wieder auf den vorher betrachteten Ausschnitt zurück. Einfacher geht's kaum.

Auch die Option "Text gadget filter" hat ihre Berechtigung. Es gibt einige Tastenkombinationen, die in Textfeldern Funktionen bewirken, anstatt in den Text übernommen zu werden, z.B. Ctrl-X zum Löschen der Zeile, die gerade editiert wird. Diese Zeichen werden sinnvollerweise ausgefiltert, denn was würde es nützen, eine Zeile mit Ctrl-X zu löschen um dann anschließend ein Ctrl-X Steuerzeichen im Textpuffer vorzufinden. Sollte jedoch einmal die Notwendigkeit bestehen, dieses oder andere verwendete Steuerzeichen in ein Textfeld zu übernehmen, so können Sie die Filterfunktion ausschalten, woraufhin selbstverständlich auch die Sonderfunktionen nicht mehr funktionieren.

Die letzten Einstellungen, überschrieben mit "Coercion", beziehen sich ausschließlich auf den Productivity-Modus. Da der Monitor diesen in einer anderen Frequenz darstellt, kann es zu Problemen kommen, wenn neben dem Productivity-Screen auch noch ein Screen in einer der normalen Auflösungen dargestellt werden soll. Während der Anzeige kann nämlich auch ein Multiscan-Monitor die Frequenz nicht ändern, was zur Folge hat, daß ein Teil des Bildschirmes verzerrt würde. Deshalb versucht der Amiga, mittels des

Interlace-Modus und unter Verlußt von Farben das Bild so gut es geht noch darzustellen. Das aber wiederum bedeutet, daß der Bildschirm plötzlich flickert und die Farben durcheinanderkommen. Diese Effekte können Sie mit "Aviod flicker" und "Preserve colors" verhindern, nehmen dafür aber möglicherweise ein umso stärker verzerrtes Bild in Kauf.

2.2.1.14 - Sound (WB 2.1 und WB 3.0)



Sogar der Piepser kann verändert oder ersetzt werden

Die nun folgenden Voreinsteller sind erst ab der Workbench 2.1 enthalten und da diese Workbench-Versionen bereits die deutsche Sprache unterstützen, werden für den Rest der Besprechung der Voreinsteller die deutschen Begriffe für Texte, Gadgets oder ähnliches verwendet. Sie kennen ja das beliebte Bildschirmblitzen, daß mangels eines internen Piepser, wie er von PCs her bekannt ist, den Anwender auf irgend etwas aufmerksam zu machen versucht. Da aber allein wegen der hervorragenden Soundeigenschaften fast jeder seinen Amiga an irgendeine Art Verstärker angeschlossen hat, kommt nun doch auch das Verlangen nach einer entsprechenden Funktion auf.

Da wir aber von einem Amiga reden, begnügen wir uns nicht mit einem lapidaren "Pieps", sondern da muß schon mehr her. Doch zunächst einmal können Sie "Anzeige blitzen" und "Ton ausgeben" getrennt voneinander ein- oder ausschalten. Haben Sie sich für einen Ton entschieden, steht die Wahl zwischen einem einfachen "Piepsen" oder aber "Digital. Sound" frei. In beiden Fällen können Sie "Lautstärke" und "Tonhöhe" mittels Schieberegler einstellen, im Falle des Piepstons auch noch die "Piepslänge". Sollten Sie aber ein gutes Sample bevorzugen (möglicherweise etwas in der Art: "He,

aufwachen!")), dann erhalten Sie über "Sound auswählen..." einen Filerequester, in dem Sie nach dem entsprechenden Filenamen suchen können. Zu guter Letzt läßt sich der Sound auch noch testen, damit Sie nicht erst in Verlegenheit kommen, den Rechner so lange quälen zu müssen, bis er freiwillig ein Bildschirmblitzen von sich gibt.

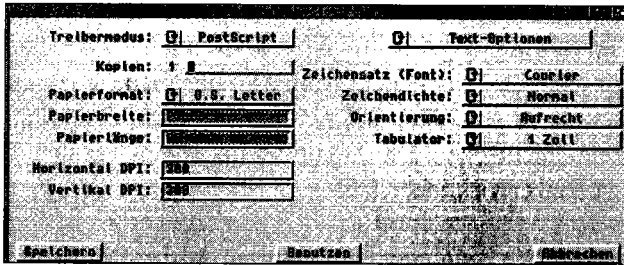
Noch ein kleiner Tip für Anwender von AmigaBASIC, das leider mit den neueren Amigas nicht mehr mitgeliefert wird - obwohl es auf einem Amiga 1200 genauso läuft, wie bei den ersten Amiga 500: Sollten Sie bisher die Erfahrung gemacht haben, daß jeder auftretende Fehler in einem Basic-Programm und der Befehl "BEEP" mit dem Absturz des ganzen Rechners quittiert wird, können Sie dem abhelfen. Der AmigaBASIC-Interpreter kommt immer bei dem Versuch, einen Pieps von sich zu geben, mit dem neuen Betriebssystem in Konflikt, sofern Sie in der Voreinstellung festgelegt haben, daß das Betriebssystem selbst piepsen oder den Bildschirm aufleuchten lassen soll - also die Gadgets "Anzeige blitzen" und "Ton ausgeben" aktiv sind. Wenn Sie diese Gadgets inaktivieren, sollte das AmigaBASIC wie gewohnt laufen.

PAG Sandini

2.2.1.15 - PrinterPS (WB 2.1 und WB 3.0)

Der Voreinsteller zu Postscript-Druckern ist im Prinzip eine Zusammenfassung von "Printer" und "PrinterGfx", bezogen auf Postscript. Die wesentlichen Elemente finden Sie auch in den genannten Abschnitten in diesem Buch erklärt. Vielleicht erwähnenswert ist die Einstellbarkeit der "Kopien", da bei Postscript-Druckern das Verfahren des wiederholten Ausdrucks ja ganz anders ist als bei anderen Druckern. Hier werden nicht nochmals alle Daten zum Drucker geschickt, was auch ziemlich aufwendig wäre, da bei Postscript ja erst einmal die komplette Seite aufgebaut werden muß, sondern die sich noch im Speicher des Druckers befindende Seite wird einfach nochmals ausgegeben.

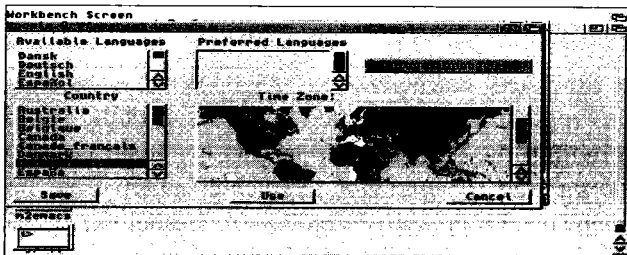
Außerdem können Sie mittels der DPI-Einstellungen noch die Auflösung des Druckers bestimmen, was in etwa dem "Density"-Regler der normalen Druckertreiber entspricht.



Wer einen Postscript-Drucker besitzt, kann diesen hiermit anpassen.

2.2.1.16 - Locale (WB 2.1 und WB 3.0)

Das Herzstück der Workbench-Systeme der neueren Generation ist die sogenannte "localization", also die Konfiguration des Systems an das gewünschte Land. Dies beinhaltet in erster Linie die Sprache, aber auch Kleinigkeiten wie Landeswährung oder das Format von Zeit und Datum. Im Moment berücksichtigt diese Einstellungen nur das Betriebssystem selbst. Doch wenn in Zukunft neue Programme auf den Markt kommen, wird es kein Nachblättern im Wörterbuch mehr geben, um den Sinn der Funktionen zu verstehen, sondern Sie werden erst gar nicht merken, daß es sich um kein deutsches Produkt handelt.



Damit Ihr Amiga deutsch spricht, stellen Sie hier die gewünschte Sprache ein.

So stellen sich Commodores Entwickler die modernen Programme zumindest vor. Inwieweit dieses System wirklich seine Erwartungen erfüllen kann, muß sich erst noch herausstellen. Denn die deutschen Übersetzungen mancher

Gadgets und Menüpunkte schafft auch nicht immer Klarheit, manchmal sind die englischen Begriffe sogar treffender und für mich auch geläufiger. Da ich schon seit gut sechs Jahren mit dem Amiga arbeite, habe ich mich schon so an seine Muttersprache gewöhnt, daß es für mich eine Umstellung wäre, die Workbench einzudeutschen. Die Zukunft wird zeigen, ob sich der unbestritten gute Gedanke verwirklichen läßt.

Doch was müssen Sie denn nun tun, um Ihrem Amiga Deutsch beizubringen. Es ist äußerst einfach. Da Sie bei der Installation der Workbench 2.1 wahrscheinlich schon die deutsche Variante - und nur diese - ausgewählt haben, sind in dem Feld "Verfügbare Sprachen" auch nur "Deutsch" und "English", das immer zur Verfügung steht, aufgeführt. Benutzer der Workbench 3.0 müssen auf diesen Komfort leider verzichten, da die Startup-Sequence keinerlei Einstellungen von Seiten des Benutzers erlaubt. So müssen Sie die notwendigen Schritte "von Hand" erledigen. Da das ganze System jedoch sehr speicherintensiv ist, rate ich ab, diese Einstellungen vorzunehmen, wenn Sie nur über das interne Diskettenlaufwerk verfügen. Optimal kann diese Möglichkeit erst dann Anwendung finden, wenn Sie ihr System auf einer Festplatte installieren. Das Installationsprogramm nimmt dann jedoch wiederum die Arbeiten ab, die nötig sind, um gleich mit dem Local-Programm arbeiten zu können - es ersetzt damit die Startup-Sequence der Workbench 2.1. Wenn Sie die Festplatte mit Hilfe des Installationsprogrammes eingerichtet haben, sollten auch jetzt in dem Feld "Verfügbare Sprachen" "Deutsch" und "English" aufgeführt sein.

Klicken Sie einfach auf "Deutsch", worauf dieses in dem Kasten "Bevorzugte Sprachen" erscheint. Bevorzugt deshalb, weil es ja immer sein kann, daß ein Programm gerade die von Ihnen ausgewählte Sprache nicht unterstützt, woraufhin die nächste Sprache aus der Liste gesetzt wird. Wird überhaupt keine der bevorzugten Sprachen unterstützt, wird in der Standardsprache des Programms fortgefahren. Deshalb ist es auch gar nicht so sinnlos, nach "Deutsch" auch noch "English" anzuwählen, denn es kann ja sein, daß ein Programm nur seine Landessprache und eben Englisch unterstützt.

Allerdings sollten die zukünftigen Programme alle möglichen Sprachen beherrschen, so daß dieser Fall eigentlich nie auftreten sollte. Um die Liste der bevorzugten Sprachen zu löschen, kann man das mit "Sprachen löschen" be-

zeichnete Gadget betätigen. Zur weiteren Wahl landesspezifischer Einstellungen, können Sie unter "Land" noch ihr Heimatland anklicken, was Ihnen in Zukunft so ungewöhnliche Datumsformate wie "02-17-92" erspart und dafür "17.02.92" angezeigt wird. Und schließlich kann noch die zutreffende Zeitzone gesetzt werden, indem Sie auf Ihrer momentane Position in der Weltkarte klicken. Da diese ein wenig klein geraten ist, sollte vielleicht darauf hingewiesen werden, daß beispielsweise Deutschland bei +1 Stunde gegenüber GMT liegt. Dummerweise wird das "+" nicht mit angezeigt, aber so weit daneben werden Sie schon nicht klicken!

2.2.2 - Utilities

2.2.2.1 - Clock

Da nun die primären Installations- und Konfigurationsarbeiten abgeschlossen sind, ist es an der Zeit, auch einmal etwas sinnvolles mit dem neuen Rechner anzufangen. Und was gibt es da für uns Digitaluhrträger besser geeigneteres als die "Clock". Wenn Sie dieses Programm also eine Zeit lang genossen haben, und fasziniert verfolgt haben, wie Ihre Lebensuhr immer weiterläuft, sollten wir vielleicht fortfahren, denn es gibt Atemberaubenderes. Es gibt auch nichts neues zu sehen, da das "Clock"-Utility breits unter Workbench 1.2 existierte. Es sei nur noch schnell auf die Menüleiste hingewiesen, die noch einige mehr oder weniger sinnvolle Optionen zur Verfügung stellt. Um die Sache noch etwas interessant zu gestalten, überlasse ich es Ihrer Experimentierfreudigkeit, die möglichen Funktionen auszutesten.

2.2.2.2 More

Mit "More" läßt sich schon eher etwas anfangen, denn das Lesen von Texten und vor allem Anleitungen ist eine der wichtigsten Arbeiten eines gewissenhaften Computeranwenders. Es bestehen zwei Möglichkeiten, dieses Programm zu starten. Entweder, Sie klicken zuerst einmal auf das File, das Sie lesen wollen (es sollte sich dabei selbstverständlich um ein Textfile handeln), drücken dann die Shift-Taste klicken dabei das "More"-Icon zweimal an. Dadurch wird der gewählte Text gleich mitgeladen.

Startet man "More" jedoch, ohne vorher ein anderes Symbol angewählt zu haben, so öffnet sich nicht nur das Textfenster sondern auch ein Filerequester, mit dessen Hilfe man den Text auswählen kann, den man gerne lesen möchte. Benutzer der Workbench 1.2/1.3 mußten da leider etwas mehr tun, denn hier gab es noch keine Filerequester, so daß per Hand eingegeben werden mußte, welcher Text geladen werden sollte. Ist diese Datei nicht im gleichen Verzeichnis wie "More" selbst, so ist auch der Pfad zu der Datei anzugeben.

Doch nun zu der neueren Version mit dem Filerequester. Es bietet sich hier auch gleich an, den Standard-Filerequester zu erklären. Denn nachdem vor AmigaOS 2.0 jeder Programmierer seinen eigenen Filerequester programmieren mußte und die Ergebnisse auch dementsprechend unterschiedlich waren, gibt es jetzt einen Standard, der auch für die Entwickler neuer Software leicht anzuwenden ist, weshalb davon ausgegangen werden kann, daß Sie sich nie wieder an einen anderen Filerequester gewöhnen müssen - vorausgesetzt natürlich, daß Sie dem Amiga treu bleiben. Da wäre zunächst einmal ein Feld, in dem das gewählte Inhaltverzeichnis erscheint. Den angezeigten Ausschnitt können Sie selbstverständlich mittels des Rollbalkens und der Pfeile am rechten Rand verschieben. Darunter gibt es ein Feld namens "Pattern", in dem ein Muster für die anzuzeigenden Filenamen angegeben ist. Dies entspricht den AmigaDOS-Konventionen und sollte bereits von älteren Betriebssystem-Versionen bekannt sein. Für Neueinsteiger sei auf den Abschnitt über das AmigaDOS hingewiesen. Die wichtigsten beiden Muster sind in diesem Zusammenhang sicherlich "#?" und "~". Ersteres ist ein Muster, das auf alles zutrifft und letzteres invertiert das Muster, d.h. es werden alle Dateien angesprochen, die nicht dem nachfolgenden Muster gerecht werden. So können mit "~(#?.info)" alle Files mit Ausnahme derer, die mit ".info" enden, angezeigt werden.

Wieder eine Zeile tiefer folgt nun der Pfadname und darunter dann der eigentliche Dateiname. Diese Anzeigen werden ständig aktualisiert, wenn Sie sich im oberen Bereich durch den Verzeichnisbaum klicken. Sie können aber auch gleich die entsprechenden Angaben über die Tastatur eingeben. Mit "OK" bestätigen Sie Ihre Wahl, was natürlich auch mittels Doppelklick auf den Filnamen geht, mit "Cancel" brechen Sie die Dateiauswahl ab. Außerdem läßt sich über "Parent" ins darüberliegende Verzeichnis zurückspringen oder mit "Drives" (oder bei der neuen Version 3.0 auch "Volumes" genannt) die Liste aller logischen Geräte anzeigen. Das sind zum einen Diskettenlauf-

werke und Festplatte, aber auch fest zugewiesene Verzeichnisse wie etwa "S:" oder "DEVS:". Näheres hierzu erfahren Sie im Abschnitt über das AmigaDOS. Sollten Sie ein Freund der Menütechnik sein, so werden auch Sie bedient, denn all diese Funktionen lassen sich auch über ein Menü erreichen. Doch nun zum "More"-Utility selbst. Dummerweise wurde immer noch keine angenehme Benutzeroberfläche eingebaut, so daß Sie hier voll auf Ihre Tastatur angewiesen sind. Alle Tastaturkürzel an dieser Stelle aufzuführen, würde jedoch den Absichten dieses Buches trotzen, weshalb nur die drei wichtigsten erwähnt seien. Mit der Return-Taste zeigen Sie die nächste Zeile an, mit der Space-Taste blättern Sie eine ganze Seite vor und mit der Backspace-Taste eine Seite zurück. Sollten Sie dieses Utility noch ausgiebiger nützen wollen, so seien Sie auf das Handbuch verwiesen.

2.2.2.3 - Display (Nur WB 2.0 und 2.1)

Dieses kleine Utility zum Anzeigen von IFF-Grafiken sei nur der Vollständigkeit halber aufgeführt, denn da keine Verbesserungen an ihm vorgenommen wurden, ist es immer noch nur sehr unhandlich zu bedienen, bei AmigaOS 3.0 fehlt es gänzlich. Der beste Weg bleibt nach wie vor, ähnlich dem "More"-Utility zuerst das Icon des Bildes, das man anschauen möchte, auszuwählen um dann mit gehaltener Shift-Taste das Display-Icon doppelzuklicken. Da außerdem die wesentlichsten Funktionen in einem kleinen Fenster angezeigt werden, werden wir erst gar nicht Ihre kostbare Zeit verschwenden, sondern gleich fortfahren.

2.2.2.4 - MultiView (WB 3.0)

Vorher jedoch möchte ich alle trösten, die bislang vergeblich auf der neuen Workbench 3.0 eines der beiden eben besprochenen Programme gesucht haben. Hier wurde der längst schon notwendige Schritt gemacht, beide Programme wurden zusammengefügt, verbessert und unter dem Namen "MultiView" auf der Workbench untergebracht. Hier steht dem Benutzer zum einen für alle möglichen Dateiformate ein File-Requester zur Verfügung, denn "MultiView" kann sowohl Text als auch Grafik und Musik verarbeiten. Dabei entscheidet das Programm selbstständig, welche Art Daten es vom Benutzer vorgesetzt bekommen hat. Eine weitere Verbesserung stellen die

Rollbalken beispielsweise bei langen Textdateien dar, da hier mit der Maus der Text gescrollt werden kann und nicht mehr wie bei "More" die Tastatur dazu gebraucht wird. Experimentieren Sie ruhig ein bißchen mit diesem Programm, so können Sie beispielsweise mal versuchen, die Startup-Sequence zu finden und zu betrachten.

2.2.2.5 - Say

Auch das Say-Utility zur Ausgabe von Sprache hat sich seit der ersten Workbench-Version 1.2 nicht verändert und verdient keine weitere Betrachtung. Noch dazu werden Sie es wohl niemals sinnvoll einsetzen können, weshalb es auch völlig sinnlos wäre, sich die ohnehin unkomfortable Bedienung einzuprägen. Lediglich ein kleiner Hinweis, falls Sie es doch einmal ausprobieren möchten und nichts funktioniert: Selbstverständlich benötigt dieses Programm den "Speak"-Treiber, der deshalb bei der Workbench 2.1 im "DOSDrivers"-Verzeichnis im "Devs"-Ordner stehen muß. Seit der Version 3.0 ist das Programm vermutlich wegen Speicherplatzmangel von den Systemdisketten verschwunden.

2.2.2.6 - Exchange (WB 2.0 und WB 2.1)

Dieses Utility bedarf schon ein wenig mehr Beachtung. Allerdings gehört es nicht ins "Utilities"-Verzeichnis und ist bei der Workbench 2.1 auch daraus entfernt worden. Denn eigentlich ist es ein "Commodity" und gehört sich auch in den entsprechenden Ordner. Da an dieser Stelle die Commodities sowieso noch nicht besprochen wurden, wird auch dieses Utility erst später im entsprechenden Rahmen behandelt werden.

2.2.3 - Tools

Einige durchaus brauchbare Hilfsprogramme sind unter dem Namen "Tools" zusammengefaßt. Die meisten waren auch schon unter der Workbench 1.2 verfügbar (so auch die drei nächsten Programme, die vor AmigaOS 3.0 jedoch noch im "Utilities"-Ordner zu Hause waren). Doch wurden Sie jetzt überarbeitet, was in den meisten Fällen nicht nur eine Anpassung an die neue

Benutzeroberfläche bedeutet. Deshalb sollten Sie neben dieser kurzen Beschreibung unbedingt auch das Handbuch zu Rate ziehen, da wir hier nicht in der Lage sind, auf die jeweiligen Besonderheiten jeder einzelnen Version einzugehen.

2.2.3.1 - Calculator

Hierüber kann man leider nicht viel berichten, er ist hier nur der Vollständigkeit halber erwähnt. Der Calculator ist ein primitiver "Taschenrechner", der nur die Grundrechenarten beherrscht. Für kurze "Zwischendurch"-Rechnungen ist er aber zu gebrauchen.

2.2.3.2 - Grafikdump

Dieses Programm befindet sich bei der Workbench 1.2 im Ordner "System", bei der Version 1.3 im "Utilities"-Verzeichnis. Ab der Workbench 2.0 wurde es in den "Tools"-Ordner verbannt.

Als Erklärung reicht eigentlich, daß der aktuelle Bildschirm als Grafik ausgedruckt wird. Es gibt dazu zwar noch ein paar Tool Types, doch möchte ich den interessierten Benutzer auf das Handbuch verweisen.

2.2.3.3 - CMD

Dieses kleine Programm ermöglicht die Umleitung der Daten, die ursprünglich an die parallele oder serielle Schnittstelle geleitet wurden, in eine Datei. Im CLJ lautet das Befehlsformat

CMD <Gerätename> <Dateiname> [OPT S|M|N].

Dabei sind als <Gerätename> entweder "serial" oder "parallel" anzugeben, "PAR:" oder "SER:" werden nicht als Gerätenamen akzeptiert. Die Option "S" bewirkt das Überspringen aller einleitenden Steuerdaten des Schreibvorganges - normalerweise wird ein Resetbefehl an den Drucker geschickt, beispielsweise dann, wenn ein Bildschirm ausgedruckt werden soll-

te. Wenn Sie "M" als Option setzen, so bleibt der Befehl so lange aktiv, bis ein "BREAK" Signal gesendet wird, ohne dieser Option wird das Programm nach einer Datenumleitung automatisch beendet. Die Option "N" aktiviert den "NOTIFY"-Modus, bei dem nützliche Bildschirmmeldungen während des Programmablaufes gesendet werden.

CMD kann auch über die Workbench gestartet werden. Dazu müssen der Geräte-Name, Dateiname und die Optionen über die TOOL TYPES (siehe Workbench-Menü "Info") eingestellt werden:

DEVICE = serial" (oder "parallel")
FILE = Dateiname (Standard: "ram:cmd_file")
SKIP = "TRUE" (Option "S" gesetzt, bei "FALSE" Option "S" nicht gesetzt)
MULTIPLE = "TRUE" (oder "FALSE" gilt analog zu Skip für die OPTION "M")
NOTIFY = "TRUE" (oder "FALSE"; auch hier analog zu oben)

Dieses Programm eignet sich beispielsweise zum Aufspüren von Programmierfehlern in eigenen Programmen, es sind aber einige andere sinnvolle Anwendungsmöglichkeiten denkbar.

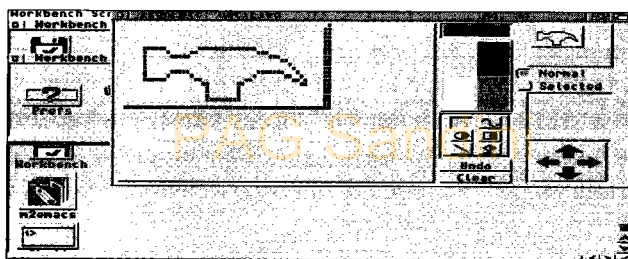
2.2.3.4 - IconEdit

Der "IconEdit" ist das Tool, mit dem Sie die hübschen kleinen Bildchen auf der Workbench erstellen und verändern können. Denn sie sollen nicht nur das Auge erfreuen, sondern auch aussagekräftig sein, und wenn Sie statt eines Papierkorbs ein schwarzes Loch bevorzugen, wie Sie es von Ihrem NEXT gewohnt sind, dann ist das durchaus machbar. Doch was ist nun das Neue an dem Editor?

Da wären zum einen die bessere Editiermöglichkeiten, die über kleine Gadgets angewählt werden können. So können Linien gezogen oder Flächen gefüllt werden und darüber hinaus der letzte Vorgang mit der "Undo"-Funktion rückgängig gemacht werden.

Weit interessanter als sie auf den ersten Blick erscheinen mögen, sind die "Normal"/"Selected"-Knöpfe, denn nun wird das erste Mal offiziell die Möglichkeit unterstützt, einem angewählten Icon ein völlig anderes Ausse-

hen zu geben, als es im nicht-angewählten Zustand hat. Und mit den beiden Knöpfen schalten Sie zwischen diesen beiden Zuständen hin und her, können also beide unmittelbar bearbeiten. Dazu müssen Sie allerdings zuvor definieren, daß es sich um solch ein Icon mit zwei Grafiken handelt. Dazu wählen Sie im "Highlight"-Menü die "Image"-Option an. Überhaupt sind die Menüs in diesem Programm recht attraktiv. Neben der Wahl des Icon-Typs und der Hervorhebung bei der Anwahl sind gerade die letzten drei Menüs beachtenswert. So verbirgt sich im "Images"-Menü ganz unscheinbar hinter der "Load"-Option ein Untermenü, das auch das Laden von IFF-Brushes erlaubt. Mit anderen Worten, Sie können die Icons mit Deluxe Paint oder einem anderen Standard-Malprogramm erstellen und anschließend im IconEdit ein Icons daraus machen, wodurch Sie natürlich weit komfortablere Bearbeitungsmöglichkeiten zur Verfügung haben, als ein solcher Editor je bieten kann.



"IconEdit3.0"

Der erste Menüpunkt aus dem "Extras"-Menü namens "Recolor" ist gerade für den Umsteiger unbezahlbar. Denn mit dieser Funktion lassen sich die Farbe zwei und drei vertauschen, was deshalb so wichtig ist, weil eben diese Vertauschung auch bei den Workbenchfarben durchgeführt wurde. Sollten Sie also bereits unter der Workbench 1.2 / 1.3 plastische Icons gezeichnet haben, so werden diese unter 2.0 bis 3.0 alle eingedrückt dargestellt. Ein kurzer Aufruf von "Recolor" genügt, und das Problem hat sich erledigt.

Nicht neu, aber dennoch erwähnenswert ist die "Use Grid?"-Option im Menü "Settings". Mit ihr läßt sich ein Netz über die Anzeige legen, wodurch die einzelnen Pixel besser abgrenzbar sind, bzw. wenn Sie diese Funktion abwählen, die tatsächliche Vergrößerung ohne irgendwelche irritierenden Spezialeffekte bekommen. Außerdem versteht der "IconEdit" eine Vielzahl von "Tool Types", die jedoch nur in den seltensten Fällen sinnvoll anwendbar sind und

im Grunde nur die Voreinstellungen für den Editor definieren. Da es aber eher unwahrscheinlich ist, daß Sie genau eine dieser Funktionen unbedingt benötigen und die Besprechung doch recht lang wäre, soll in diesem Rahmen darauf verzichtet werden. Sie finden alle Parameter im Handbuch umfassend erklärt.

2.2.3.5 - *ME*macs

MicroEMACS ist auch schon ein alter Bekannter und kann immer noch nicht mit professionellen Texteditoren mithalten. Allerdings bekam er wenigstens inzwischen umfangreiche Menüs verpaßt, so daß sich mit ihm inzwischen anständig arbeiten läßt. Die Funktionsvielfalt ist sogar schon so überwältigend, daß auf dieses Tool im Rahmen dieses Buches im Abschnitt 3.7 ausführlich eingegangen wird. Da Sie aber früher oder später einen Texteditor brauchen werden, sollten Sie sich mal mit Ihm vertraut machen. Zu mehr als zum bloßen Texte editieren ist der Editor jedoch ungeeignet. Der wohl größte Kritikpunkt ist, daß auch nach der Einführung eines Standard-Filerequester, das Programm keine Möglichkeiten bietet, Dateien in ähnlich komfortabler Weise auszuwählen. Aufgepeppt mit ein paar weiteren Features wäre dieses Programm durchaus empfehlenswert.

Ich selbst habe die Texte zu diesem Buch mit einem nahen Verwandten von ME_{macs} getippt. Ich kann mich mit der Maus nicht so recht anfreunden ("Bis ich das gute Ding unter dem Stapel Bücher, Zeitschriften, Aufzeichnungen usw. ausgegraben habe..."), so ist es für mich optimal, daß jeder Menüpunkt auch durch irgendwelche Tastenkombinationen aufgerufen werden kann ("...habe ich die wichtigsten Tastaturkürzel auswendig gelernt!"). Am besten bilden Sie sich eine eigene Meinung zu diesem Editor, Sie werden sowieso nicht an einem Texteditor vorbeikommen.

2.2.3.6 - *Colors* (WB 2.0 und 2.1)

Wenn Sie sich jetzt vielleicht fragen werden: "Was, schon wieder ein Programm zur Einstellung der Bildschirmfarben - dafür gibt es doch schon "Palette" in den Prefs, was soll das?", dann muß man Ihnen auf den ersten Blick recht geben, denn auch mit "Colors" läßt sich - ebenso wie mit "Palette"

- die Farbtabelle eines Bildschirmes verändern. Aber eben "eines" Bildschirms, und zwar eines beliebigen und nicht nur die des Workbench-Screens. Denn "Colors" kann sein Fenster auf jedem beliebigen Screen öffnen - was auch ein neues Feature seit AmigaOS 2.0 ist. Doch wahrscheinlich ist hierfür ein Beispiel angebracht.

Starten Sie "MEMacs". Dieser wird seinen eigenen Screen öffnen, den Sie nun ein gutes Stück nach unten ziehen, so daß die Workbench wieder sichtbar wird. Dort starten Sie nun "Colors" und dieses öffnet sein Window nun nicht auf dem Workbench-Screen, sondern auf dem von MEMacs, oder allgemein gesagt, auf dem vordersten. Jetzt können Sie den Screen wieder hochziehen, im "Colors"-Fenster die Farben nach Ihren Vorstellungen editieren und das Programm wieder verlassen.

Der Vorteil liegt auf der Hand: Auch wenn ein Programm nicht die Möglichkeit bietet, seine Farben zu verändern, so genügt ein Doppelklick und Sie haben keine Probleme mehr. Andererseits braucht ein Programmierer nicht unbedingt seinen eigenen Palette-Requester einbauen, obwohl das natürlich immer noch die komfortabelste Möglichkeit bleibt und das "Colors"-Tool nur ein Hilfsmittel für den Notfall sein kann.

Leider ist dieses nette kleine Programm von der Workbench 3.0 wieder verschwunden - möglicherweise hat es die Umstellung auf 16,7 Millionen Farben nicht verkraftet.

2.2.3.7 - InitPrinter

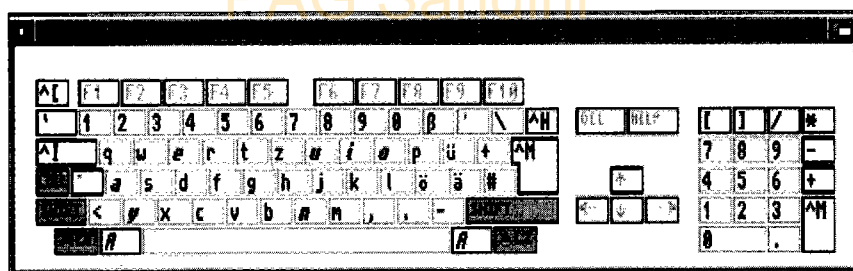
Nach einer kleinen Irrfahrt durch das Verzeichnis "System" auf der Workbench 1.2 / 1.3 hat das Programm nun endgültig sein Zuhause gefunden - im Verzeichnis "Tools".

Je nachdem, welchen Drucker Sie besitzen und in welcher Form Sie Ihren Ausdruck bevorzugen, stellen Sie in den Printer-Preferences Ihre Wünsche ein. Und diese Daten werden vor dem ersten Ansprechen des Druckers an diesen übermittelt. Wenn Sie jedoch nach diesem ersten Ausdruck den Drucker zwischenzeitlich abschalten und anschließend wieder mit Ihren gewohnten

Einstellungen arbeiten wollen, so müssen diese Daten nochmals übertragen werden. Und genau das passiert, wenn Sie das Programm "InitPrinter" starten. Diese Verfahrensweise ist weit flexibler, als wenn die Daten beispielsweise vor jedem Ausdruck erneut gesetzt werden würden, da Sie ja vielleicht zwischendurch auf andere Einstellungen umschalten möchten - beispielsweise mit einem Schalter am Drucker. Denn dann könnten derartige Eingriffe "von außen" überhaupt keine Einfluß nehmen.

Mit dem "InitPrinter"-Programm liegt es in Ihren Händen, wann der Drucker zurückgesetzt wird. Solange Sie den Drucker eingeschaltet lassen und kein anderes Anwendungsprogramm die Konfiguration ändert, brauchen Sie selbstverständlich nicht jedesmal "InitPrinter" aufrufen, da die Daten für diesen Zeitraum im Drucker gespeichert sind.

2.2.3.8 - KeyShow



"KeyShow3.0"

Das "Key Show"-Programm ist nicht für den täglichen Gebrauch geschaffen, sondern dient mehr der Information: Es zeigt die momentane Tastaturbelegung an. Dazu wird ein Abbild der Tastatur auf dem Monitor angezeigt und alle Tasten mit den ihnen zugeordneten Buchstaben dargestellt. Durch Klick auf die Ctrl- oder eine der Shift- oder Alt-Tasten können Sie sich zudem anzeigen lassen, welche Zeichen mit den entsprechenden Umschalttasten erscheinen.

Dabei ist das Programm so intelligent, daß das Aussehen der Tastatur abhängig von den verwendeten Einstellungen richtig gewählt wird (die amerikani-

sche Tastatur hat eine größere Return-Taste, dafür fehlt die Taste, die bei der deutschen Tastatur mit dem “#” belegt ist). Wenn die Taste einen Steuercode enthält, so beginnt die Anzeige mit “~” oder “^”. Sollte mehr als nur ein einziger Buchstabe auf einer Taste liegen - was durchaus möglich ist -, so weist ein “\$\$” darauf hin.

Daß dieses Programm aber durchaus seine Berechtigung hat und nicht nur eine bloße Spielerei ist, macht vielleicht folgendes Beispiel deutlich: Gerade in den Anfangszeiten des Amigas, als es nur recht vereinzelte Amiga-Anwender gab, und die Kommunikation deshalb sehr eingeschränkt war, wurde selbst in Fachzeitschriften mehrfach die Frage behandelt, wo denn der Apostroph auf der Tastatur zu finden sei. Denn das Ding, das sich ganz links oben - unterhalb der Esc-Taste - befindet, ist leider spiegelverkehrt.

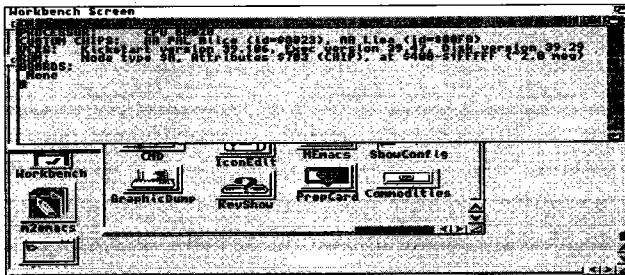
Und gerade für die damals noch scharenweise vorhandenen AmigaBASIC-Anwender war diese Taste sehr wichtig, denn Sie entspricht dem “REM”-Befehl - und abgetippte Programme, in denen die falsche Taste verwendet wurde, liefen einfach nicht. Mit dem “KeyShow”-Programm wäre das alles kein Problem gewesen, Sie hätten nach kurzer Zeit den Apostroph auf der Kombination Alt-ä orten können. Und auch wenn gerade dieses Zeichen inzwischen hinreichend bekannt sein dürfte, so sucht man doch immer wieder nach irgend welchen Exoten, wie etwa “3/4”. Und da ist “KeyShow” dann schon eine enorme Hilfe.

2.2.3.9 - PrintFiles

Dieses Programm zählt zweifellos zu einem der unentbehrlichen Utilities für den Workbench-Anwender. Wenn immer Sie ein oder mehrere Textfiles ausdrucken wollen, klicken Sie diese(s) einfach an und Doppelklicken anschließend bei gedrückter Shift-Taste das Symbol “PrintFiles”. Zur Auswahl mehrerer Files benützen Sie die erweiterte Auswahl (entweder die Shift-Taste oder das Gummiband).

Da es oftmals sehr wünschenswert ist, zwischen zwei aufeinanderfolgenden Texten einen Seitenvorschub einzufügen, können Sie in den Tool Types zu “PrintFiles” den Parameter “FLAGS=formfeed” setzen, der genau das bewirkt.

2.2.3.10 - ShowConfig



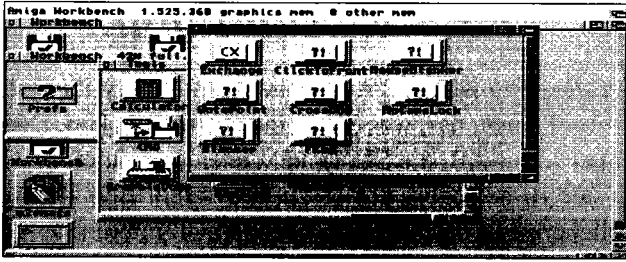
“ShowConfig3.0”

Dieses Programm gibt Ihnen die Konfigurationsdaten Ihres Rechners aus. Dazu zählen der Prozessor, die Custom-Chips, die Versionen der Kickstart, der RAM-Speicher sowie die eingebauten Erweiterungskarten. Da Sie wohl meistens wissen, welche Karten Sie in Ihren Rechner gesteckt haben und die Informationen hierzu sehr spärlich sind, helfen diese nicht sonderlich weiter.

Wesentlich interessanter sind da schon die Custom-Chip Revisionen, denn nun brauchen Sie nicht extra den Rechner aufschrauben, nur um herauszufinden, ob Sie nun schon ECS-Denise, AAA-Chipset oder nichts davon besitzen - ebenso interessant ist nun natürlich auch die Frage, welches Herz in unserer Freundin schlägt. Und da sowieso nicht jeder mit den auf den Chips aufgedruckten Nummern etwas anfangen kann, ist diese Anzeige höchst willkommen. Bevor Sie nun also in den nächsten Laden laufen, um sich einen neuen Chip zu leisten, starten Sie dieses Programm und überprüfen Sie, ob das Teil vielleicht nicht schon in Ihrem Rechner steckt!

2.2.4 - Der "Commodities"-Ordner

Die “Commodities” sind ebenfalls im “Tools”-Verzeichnis anzutreffen, haben aber ihre eigene Schublade und sind auch vom Prinzip her etwas ganz anderes als die Tools. Was ein Commodity erst zum Commodity macht, ist die Eigenschaft, daß es sich in den Input-Handler hängt und somit alle möglichen Eingaben schon vor dem Rest des Computers bekommt. Diese kann es dann nach belieben weiterverarbeiten oder auch (modifiziert) weiterleiten.



Die "Commodities3.0"

Sie können beliebig viele Commodities starten, die dann alle im Hintergrund laufen, doch bedenken Sie immer, daß abhängig davon, was das Programm macht und wie gut es programmiert wurde, einen mehr oder weniger ins Gewicht fallenden Anteil der Rechnerleistung verbraucht. Mit anderen Worten: Man kann alles übertreiben, auch die Nutzung von Commodities.

Überlegen Sie sich, welche Sie wirklich brauchen und verzichten Sie auf die restlichen. Da ja alle Commodities im Input-Handler hängen und nacheinander die einlaufenden Daten untersuchen und weiterreichen, ist es klar, daß ihre Reihenfolge äußerst entscheidend ist. Stellen Sie sich vor, zwei Commodities warten auf eine ganz bestimmte Tastenkombination, die sie anschließend anschießen. Das zweite Commodity würde bis in alle Ewigkeit warten! Deshalb kann man bei allen Commodities in den "Tool Types" unter CX_PRIORITY die Priorität bestimmen. Dadurch kann dann eine eindeutige Ordnung erstellt werden.

2.2.4.1 - *Exchange*

Das Programm "Exchange" ist das Steuerprogramm der Commodities. Sie können diese zwar nur durch einen Doppelklick starten, dann jedoch werden sie in die Liste verfügbarer Commodities aufgenommen und Sie können sie weiter bearbeiten. So werden Ihnen zum einen zusätzliche Informationen zu einem angewählten Commodity angezeigt. Wenn dieses ein eigenes Window zur Verfügung stellt (um beispielsweise weitere Parameter anzugeben), so kann es geöffnet bzw. geschlossen werden. Außerdem kann es vorübergehend inaktiv geschaltet oder ganz entfernt werden, was natürlich auch durch erneuten Doppelklick auf das Commodity-Icon selbst erfolgen kann.

2.2.4.2 - AutoPoint

Da Sie ja den standardmäßigen "AutoPoint" von Ihrer SUN-Station vermissen, ist diese Tool genau das, was Sie immer vermißt haben. Doch im Ernst: Es gibt Rechner (eben z.B. die SUN), bei denen es genügt, den Mauszeiger über ein Fenster zu bewegen, um es zu aktivieren. Beim Amiga wird ein Fenster ja üblicherweise erst aktiviert, nachdem in sein Inneres geklickt wurde. Mit "AutoPoint" schalten Sie die alternative Verfahrensweise ein. Hier ist auch sehr gut das Vorgehen der Commodities zu erkennen. Es fängt die Mausbewegung ab, und wenn sich der Mauszeiger über einem neuen Fenster befindet, wird dieses aktiviert. Natürlich werden die Mausdaten ebenso wie alle anderen anschließend weitergeleitet.

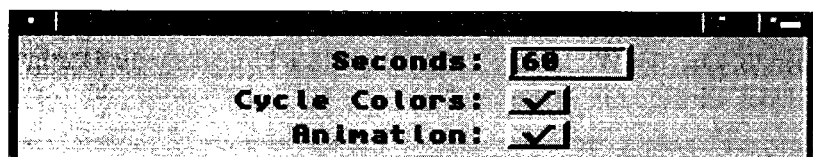
2.2.4.3 - ClickToFront

"ClickToFront" hat ebenfalls etwas mit Fenstern zu tun. Gerade bei der intensiven Nutzung vieler Windows gleichzeitig kommt es oftmals vor, daß man ein Fenster im Vordergrund haben will, dessen Vorder-/Hintergrund-Symbol unglücklicherweise gerade überlagert ist. Dann hilft "ClickToFront". Denn wenn es gestartet wurde, genügt ein Doppelklick irgendwo in das Fenster (nur nicht gerade auf ein Icon), um dieses nach vorne zu holen. Falls Sie diese Funktion jetzt gleich gierig ausprobieren wollen und nichts funktioniert, werfen Sie nicht gleich die Flinte ins Korn, sondern schauen Sie sich einmal die Tool Types von "ClickToFront" an: Da steht u.a. "QUALIFIER=irgendwas". Und wenn dieses "irgendwas" nicht gerade "NONE" ist, so bedeutet das, daß Sie zu dem Doppelklick auch noch eine bestimmte Taste gedrückt halten müssen. Zur Auswahl stehen hierbei : LEFT_ALT, RIGHT_ALT und CTRL. Eine dieser Tasten zusätzlich zum Doppelklick zur Bedingung zu machen, ist gar keine so schlechte Idee, dann holen Sie wenigstens kein Fenster unbeabsichtigt nach vorne. Andererseits bedeutet das natürlich auch wieder ein gewisses Maß an Mehraufwand - wofür Sie sich letztlich entscheiden, ist Geschmacksache.

2.2.4.4 - Blanker

Sicherlich das Commodity überhaupt ist der "Blanker". Wie Sie ja sicher wissen, sind Monitore recht empfindliche Geräte, die es auf den Tod nicht ausste-

hen können, wenn über längeren Zeitraum eine helle Grafik dargestellt werden soll. Diese kann nämlich in die Beschichtung des Bildschirms einbrennen, da diese ja unablässig mit Elektronen beschossen wird. Um dem vorzubeugen, sollte das Bild bei längerer Nichtbenutzung dunkel geschaltet werden, z.B. wenn Sie sich mal eben eine Tasse Kaffee holen. Den Monitor auszuschalten, wäre auch falsch, denn häufiges An- und Abschalten verringert die Lebensdauer eines jeden elektronischen Gerätes.



"Blanker3.0"

Und eben deshalb ist der "Blanker" das ideale Commodity. Er erlaubt es, einen Zeitraum vorzugeben, nach dessen Ablauf der Bildschirm schwarz geschaltet wird, wenn inzwischen keine Taste betätigt und die Maus nicht bewegt wurde. Denn auf diese Weise erkennt der Rechner automatisch, daß Sie ihn im Moment nicht benötigen, und kann entsprechend reagieren. Ab der Workbench 2.1 wurde der der Blanker sogar etwas erweitert; nun können Sie optional auch noch farbige Linien über den Bildschirm tanzen lassen, solange der Bildschirm schwarz ist, oder aber den kompletten Bildschirm farbig pulsieren lassen. Das zeigt Ihnen zum einen, daß der Monitor noch nicht abgeschaltet ist und zum anderen ist es ganz einfach recht nett anzusehen.

2.2.4.5 - NoCapsLock

Bei diesen Commodity geht es um die Tastatureingabe. Es bewirkt, daß die Caps-Lock Taste (das ist die mit dem LED), die normalerweise auf Großschreibung umschaltet, inaktiv wird. Die Rechtfertigung für dieses Programm ist, daß Sie nicht mehr aus Versehen die Caps-Lock Taste betätigen, und im Folgenden alles groß geschrieben werden würde. Da man aber ebenso jede andere Taste aus Versehen betätigen kann, und dieser Fehler wohl weniger auffällt als eine komplett groß geschriebene Textzeile, kann auf dieses Tool wohl durchaus verzichtet werden.

2.2.4.6 - I Help (nur WB 2.0)

“IHelp” gibt der Tastatur hingegen zusätzliche Funktionen, anstatt bestehende zu deaktivieren. Mit diesem Programm lassen sich nämlich alle möglichen Funktionen, die normalerweise mittels Maus angewählt werden, auf die Tastatur legen. Damit wird es wieder einmal noch einfacher, den Amiga nur über Tasten zu bedienen. Zwar ist die Maus ein phantastisches Hilfsmittel und eine der Hauptgründe, warum das Arbeiten mit dem Amiga so intuitiv vonstatten gehen kann, wenn man jedoch ohnehin ständig Tastatureingaben machen muß, kann der Wechsel zwischen den beiden Eingabemedien lästig werden. Doch wie funktioniert das nun?

Zunächst einmal muß darauf hingewiesen werden, daß diese Commodity unter der Workbench 2.1 und 3.0 nicht mehr existiert, da seine Funktion in dem neuen “FKey” integriert wurde (siehe auch dort). Als Workbench 2.0-Benutzer muß man wieder einmal den Umweg über die Tool Types einschlagen. Dort sind eine Reihe von Funktionen aufgeführt, wie etwa “CYCLE=” oder “MAKEBIG=”, die Sie auf eine Taste legen können, indem Sie deren Bezeichnung hinter das “=” setzen. Welche Kombinationen für die Wahl der Tasten möglich sind, entnehmen Sie bitte der weiter untenstehenden Tabelle “Tastenkombinationen”. Die einzelnen Schlüsselwörter bedeuten dabei folgendes:

CYCLE	blättern in Anwendungsfenstern
MAKEBIG	aktives Fenster auf maximale Größe bringen
MAKESMALL	aktives Fenster auf minimale Größe verkleinern
CYCLESCEEN	durch die Bildschirme blättern
ZIPWINDOW	aktives Fenster zoomen (entspricht dem Zoom-Gadget)

2.2.4.7 - FKey (WB 2.0)

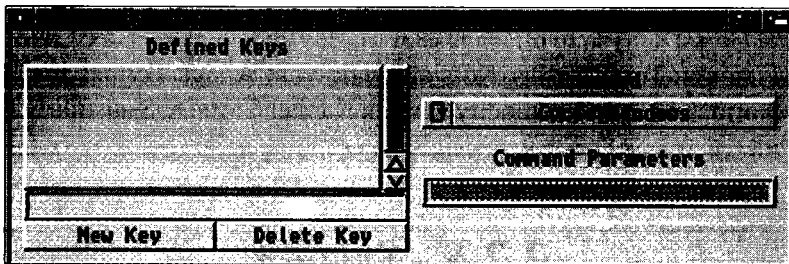
Auch “FKey” belegt die Funktions-Tasten (was auch der Grund ist, warum bei der Workbench 2.1 und 3.0 für beide Funktionen nur noch ein Programm existiert). Es erlaubt die Belegung der Funktionstasten mit Texten. Wenn Sie es starten, wird ein übersichtliches Fenster geöffnet, in dem Sie zu jeder F-Taste einen beliebigen Text eingeben können. Am unteren Ende ist noch ein mit “Modifier” betitelttes Gadget, bei dem Sie noch auswählen können, ob zu-

sätzlich zur Funktionstaste auch noch die Shift-Taste gehalten werden muß, so daß es insgesamt möglich ist, 20 Tasten zu belegen (je 10 mit und ohne der Shift-Taste). Das Ganze läßt sich dann noch abspeichern um die Eingaben dauerhaft zu machen.

2.2.4.8 FKey (WB 2.1 und 3.0)

Wie bereits erwähnt, sind die Funktionen des "IHelp"-Commodity in das neue "FKey" integriert worden. Von nun an können sowohl Texte als auch Funktionen auf jede beliebige Taste gelegt werden. Auch die umständliche Handhabung mit den Tool Types entfällt. Es wird ein sauberes Fenster geöffnet, in dem Sie eine Liste der belegten Tasten angezeigt bekommen. Mit "Taste dazu" fügen Sie eine neue Taste hinzu, mit "Taste löschen" entfernen Sie eine vorher definierte.

Nachdem Sie die Bezeichnung der Taste eingetippt haben (siehe Tabelle "Tastenkombinationen"), können Sie mit dem unter "Befehl" stehenden Gadget die Funktion der Taste auswählen. Dabei stehen die bereits unter "IHelp" erwähnten Funktionen zur Verfügung, ebenso wie "Text einfügen", was dem bisherigen "FKey" der Workbench 2.0 entspricht und zwei neue Varianten: "Programm starten" und "ARexx-Skript starten". In beiden Fällen gehört der zugehörige Filename in das Text-Gadget "Befehlsargumente". Auf diese Weise können Sie ganze Programme mit nur einer einzigen Taste starten - die schnellste Möglichkeit überhaupt.



"FKey3.0"

Um die letzten Programme auch sinnvoll anwenden zu können, ist es nötig zu wissen, wie man die Tasten eigentlich korrekt bezeichnet.

Dabei soll Ihnen folgende Tabelle eine Hilfe sein:

Tastenkombinationen

Alt	eine von beiden
Alt-Tasten RAlt	die rechte Alt-Taste
LAlt	die linke Alt-Taste
Shift	eine der beiden Shift-Tasten
RShift	die rechte Shift-Taste
LShift	die linke Shift-Taste
RCommand	die rechte Amiga-Taste
LCommand	die linke Amiga-Taste
Control	die Ctrl-Taste
Numericpad	eine Taste vom abgesetzten Zehnerblock
RightButton	der rechte Mausknopf
MiddleButton	wer ihn besitzt: der mittlere Mausknopf
LeftButton	der linke Mausknopf

Diese Tasten können natürlich auch miteinander kombiniert werden und meistens kommt auch noch ein Buchstabe, eine Ziffer oder F-Taste hinzu. Beispiele hierfür sind: "LAlt b", "Control Shift F1" oder "LeftButton x".

2.2.4.9 - CrossDOS

Dieses Hilfsprogramm kann nur in Verbindung mit den Treibern PC0: und PC1: (siehe auch das Kapitel zu dem Ordner "Devs" und "Storage" und das Kapitel CrossDOS) sinnvoll eingesetzt werden und dient in erster Linie zum Anpassen von Texten bei der Konvertierung vom Amiga-Format zum MS-DOS-Format. Dieses Commodity wird im Kapitel CrossDOS ausführlich behandelt.

2.2.5 - System

Auch im "System"-Ordner sind noch einige Utilities enthalten, die wohl nicht so ganz in den "Utilities"-Schubladen gepaßt haben. Dennoch sind es Utilities, die Sie ebenso gebrauchen können, wie alle anderen.

2.2.5.1 - NoFastMem

“NoMemFast” begleitet den Amiga nun auch schon eine Zeit lang. Falls Sie immer noch nicht wissen, was es tut: Es belegt den gesamten Fast-RAM (und gibt ihn bei einem erneuten Aufruf wieder frei). Nötig ist dies eigentlich nur bei sehr alten und unsauber geschriebenen Programmen, die davon ausgehen, daß nur Chip-RAM vorhanden ist. Da solche Programme aber einerseits schon längst durch neuere verbesserte Versionen ersetzt worden sein sollten und andererseits ein Programm, das nicht einmal Fast-RAM verkraftet, wohl kaum eine höhere Kickstart-Version als 1.3 verkraftet, ist die Anwendung dieses Programmes ab AmigaOS 2.0 doch recht zweifelhaft. Wenn Sie also nicht zufälligerweise ein unverzichtbares Programm besitzen, daß nur nach dem Start von “NoFastMem” funktioniert, dann vergessen Sie es!

2.2.5.2 - SetMap (WB 1.2 / 1.3 / 2.0)

Nur noch auf der Workbench 2.0 gesichtet - seitdem jedoch verschollen - ist “SetMap”, denn ab der Workbench 2.1 gehört die Tastaturbelegung ja bereits zu den Voreinstellern und wird dementsprechend mit einem Preferences-Utility gesetzt. Da die richtige Tastaturbelegung sowieso beim Systemstart von der “Startup-Sequence” geladen werden sollte, ist dieses Programm nur sehr selten nötig. Sollten Sie aber dennoch zwischendurch auf eine andere Tastaturbelegung umschalten wollen, so geben Sie in den Tool Types einfach “KEYMAP=” und die Ländererkennung an. Zur Auswahl stehen hier alle in DEVS:Keymaps vorhandenen Tastaturbelegungen wie etwa “d” (deutsch), “f” (französisch) oder die standardmäßige “usa” und andere.

2.2.5.3 - DiskCopy (WB 1.2 / 1.3 / 2.0)

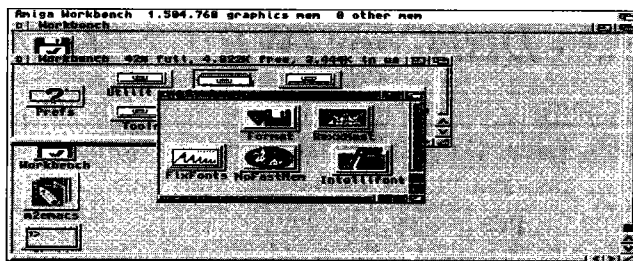
Das Programm “DiskCopy” wird auch nur noch unter der Workbench 2.0 über Icon aufgerufen. Da diese Funktion ab der Workbench 2.1 in den “Copy”-Befehl aus dem “Icons”-Menü integriert wurde, sollten Sie sich wenn nötig den entsprechenden Abschnitt bei der Besprechung des Menüs durchlesen. Es gibt also nicht mehr die Möglichkeit, durch anklicken eines Icons eine Diskette zu kopieren.

2.2.5.4 Format

Auch "Format" ist ja nichts neues, aber es ist dennoch nach wie vor notwendig. Mit ihm formatiert man Disketten oder auch andere Speichermedien und macht Sie somit nutzbar. Bis zu der Workbench Version 2.0 mußte man nach wie vor erst ein Diskettensymbol anklicken, bevor man durch Doppelklick bei gedrückter Shift-Taste auf das Format-Icon das Programm und damit den Formatiervorgang startet. Die neue Version, die mit der Workbench 2.1 Einzug gehalten hat, zeichnet sich hingegen durch die aufgepeppte Benutzeroberfläche auf, die nun die Auswahl aus der Liste möglicher Devices zuläßt.

Dabei erhält man zusätzlich Informationen über die Kapazität des Speichermediums und zu welchen Anteil es vorher belegt war. Nachdem man das gewünschte Device ausgewählt hat, können noch weitere Angaben zur Formatierung selbst gemacht werden. Da ist natürlich zuerst einmal der Name der Diskette, der nun nicht mehr nachträglich mittels eines "Rename"-Aufrufes umbenannt werden muß. Darüber hinaus läßt sich bestimmen, ob ein Trashcan eingerichtet und ob das FastFilesystem benutzt werden soll. In letzterem Fall ist es nicht mehr möglich, auf die Diskette mit einem Betriebssystem vor 2.0 zuzugreifen. Daran sollten Sie denken, wenn Sie die Diskette weiterreichen möchten. Andererseits passen auf eine FFS-Diskette mehr Daten und der Zugriff erfolgt auch merkbar schneller.

Gestartet wird die eigentlich Formatierung dann mittels Auswahl des Gadgets "Formatieren" für eine volle Formatierung, oder auf "Schnell", für die schnelle Formatierung von Disketten, die schon einmal im AmigaDOS-Format beschrieben wurden.



"System 3.0"

Mit Einführung der Workbench 3.0 wurde dieses Programm weiterentwickelt, so daß es weitere Optionen anbietet. So können Sie jetzt angeben, ob Sie den internationalen Modus benutzen wollen, bei der Probleme bei der Groß-/Kleinschreibung internationaler Zeichen gelöst werden sollten. Da dieser Modus aber ebenfalls nicht von Amigas mit älteren Kickstartversionen als 2.0 verstanden wird, gelten die oben ausgesprochenen Warnungen bezüglich des FFS hier entsprechend.

Die letzte Option wird sogar nur von den neuesten Amiga-Modellen unterstützt, so daß nicht einmal AmigaOS 2.0 was damit anfangen kann: Mit dem Directory-Cache wird das Öffnen von Schubladen, Dateiauswahlfenstern und Listen beschleunigt.

2.2.5.5 - *RexxMast* (WB 2.0 / 2.1 / 3.0)

Ein weiteres Geheimnis der Workbench 2.0 verbirgt sich hinter dem Programm "RexxMast". Da der Begriff "ARexx" inzwischen schon mehrfach durch alle Zeitschriften gegangen ist und wohl jeder nur halbwegs interessierte Anwender zumindest eine grobe Vorstellung davon hat, worum es dabei geht, traut man sich kaum, es noch einmal auszusprechen: ARexx ist eine Programmiersprache, die in besonderem Maße die Interaktion von Programmen unterstützt. Soweit so gut. Also praktisch eine Art BASIC, das von mehreren Anwendungen angesprochen werden kann.

Mehr zu ARexx finden Sie im Kapitel „ARexx für Einsteiger“ am Ende des Buches, weshalb wir uns an dieser Stelle sehr kurz halten. Egal was ARexx nun letztendlich wirklich ist und was man damit genau anfangen kann, wesentlich ist, daß Sie zuerst einmal RexxMast starten müssen, bevor Sie darauf zugreifen können. Es läuft dann unscheinbar im Hintergrund und springt erst dann an, wenn ARexx wirklich genutzt wird. Ohne dem "RexxMast" jedoch läuft auch gar nichts. Wer also häufiger mit ARexx arbeiten möchte, der sollte den "RexxMast" auf jeden Fall gleich beim Start der Workbench automatisch aktivieren lassen. Wenn Sie allerdings nur selten von den Qualitäten von ARexx Gebrauch machen wollen, genügt es, das Programm über sein Gadget im System-Ordner zu starten. So sparen Sie ein klein wenig Speicher, da das Programm ja resident gehalten werden muß.

2.2.5.6 - FixFonts (WB 2.0 / 2.1 / 3.0)

Um zu verstehen, was "FixFonts" eigentlich macht, muß ein klein wenig auf die Struktur der Amiga Font-Daten eingegangen werden. Zunächst einmal liegen Fonts immer im "FONTS:"-Verzeichnis. Dort existiert zu jedem Font eine Datei mit dem Namen des Zeichensatzes plus der Endung ".font" und ein eigenes Directory, das ebenfalls den Namen des Zeichensatzes trägt und in dem die eigentlichen Daten gespeichert werden. Diese sind dann in Files gespeichert, die die Höhe des Zeichensatzes als Namen tragen.

Um ein Beispiel zu nennen: Der Opal-Font hat ein File "Opal.font" und ein Verzeichnis "Opal" mit den Dateien "9" und "12". Das File "Opal.font" enthält dabei einige Daten, die den Font betreffen und auch noch die zur Verfügung stehenden Höhen, im Beispiel also die Werte "9" und "12". Was passiert nun, wenn Sie das File "9" löschen? Das File ist dann weg, die "9" steht aber immer noch im "Opal.font". Umgekehrt stimmen die Daten auch nicht mehr überein, wenn nachträglich eine neue Höhe hinzukopiert wurde. Und um genau diese Falschinformationen in den ".font"-Files zu korrigieren, gibt es das Programm "FixFonts". Es durchsucht das komplette Fonts:-Verzeichnis und seine Unterverzeichnisse und ändert die Einträge in den ".font"-Dateien entsprechend. Sollten Sie also jemals Änderungen Ihrem Font-Ordner durchführen, rufen Sie einfach hinterher "FixFonts" auf und alles ist wieder in Ordnung.

2.2.5.7 - Fountain (WB 2.0 und WB 2.1)

Ebenfalls mit den Fonts hat das Programm "Fountain" zu tun. Neben der oben erwähnten standardmäßigen Vorgehensweise der Bitmap-Fonts mit fest definierten Höhen wird seit der Workbench 2.0 auch das Konzept der Vektorfonts unterstützt. Vektorfonts sind Zeichensätze, die nicht als Grafikelemente abgespeichert sind, sondern als Berechnungsvorschriften. Da heißt es dann z.B. "mache einen geraden Strich und verbinde die beiden Enden mit einem Bogen auf der rechten Seite und du erhältst ein großes D". Derartige Informationen können dann natürlich unabhängig von der Größe in gleichbleibend guter Qualität dargestellt werden, was gerade in Verbindung mit der

Druckerausgabe eine große Rolle spielt, da die Auflösung von Druckern im Normalfall höher ist als die von Monitoren, weshalb dort die Buchstaben größer sein müssen. Doch jetzt zu "Fountain" selbst. Alle Gadgets ausführlich zu beschreiben, wäre an dieser Stelle wohl ein wenig zuviel des Guten, denn die Bedienung ist auch so recht einleuchtend, wenn man weiß, was man mit dem Programm anfangen kann. Bevor Sie die Fonts in beliebiger Größe benutzen, müssen Sie zuerst berechnet werden, und diesen Teil übernimmt "Fountain". Dabei geben Sie einfach den gewünschten Font und dann die Größe an, die Sie gerne hätten. Außerdem ist es möglich, in der gewählten Größe auch einen Bitmap-Font berechnen zu lassen, was die Textausgabe erheblich beschleunigt, da ja nun keine weiteren Berechnungen bei der Anzeige selbst mehr nötig sind. Andererseits kostet das natürlich wieder Speicherplatz auf der Diskette. Als Kompromiß sollten Sie sehr häufig verwendeten Größen zu Bitmap-Fonts vorberechnen lassen, weniger gebräuchliche nur definieren.

Ein ähnliches Programm ist auch auf der ExtrasD der Workbench 3.0, ebenfalls in der System-Schublade zu Hause und heißt "IntelliFont".

PAG Sandini

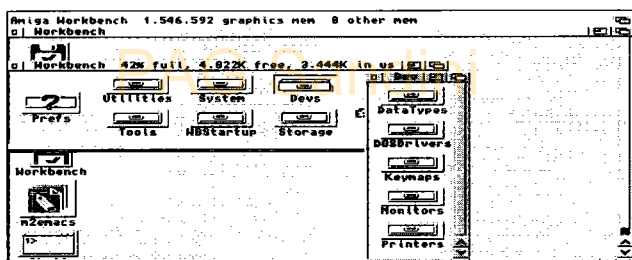
2.2.5.8 - CLI (WB 1.2/1.3/2.0) und Shell (1.3/2.0/2.1/3.0)

Zweimal das gleiche unter einem anderen Namen ist die Schnittstelle zum "Command Line Interface" (CLI), das inzwischen (seit 2.0) eine "Shell" aufweist. Vor der Kickstartversion 2.0 haben sich beide sehr stark unterschieden, die Shell war wesentlich komfortabler, wenngleich beide den gleichen Zweck erfüllten.

Prinzipiell wird das Fenster, das Sie mit dem netten kleinen "Prompt" zur Eingabe irgendwelcher Befehle auffordert, "CLI" genannt. Da aber im Zuge der Weiterentwicklung auch schon solche Feinheiten wie editierbare Zeilen hinzugekommen sind, wird das ganze jetzt unter der wohlklingenden Bezeichnung "Shell" angepriesen, was jedem Computerbesitzer das Gefühl von Sicherheit und Wohlstand auf Tastaturebene vermittelt. Doch mal ganz im Ernst: Das "CLI" der Workbench 2.0 unterscheidet sich in keinsten Weise von der "Shell" der Workbench 2.1 oder 3.0.

Was uns aber letztendlich interessiert, ist ja nichts weiter, als was wir mit dem Ding anfangen können. Und um diese Frage zu beantworten, müssen Sie sich vergegenwärtigen, daß Sie mit der Shell nicht eine Art "Utility" starten, also nicht eine Ebene nach oben klettern, sondern vielmehr eine Stufe hinabsteigen, zu der dem Rechner näher stehenden Kommando-Ebene. Hier können dann die Fähigkeiten des Amigas voll ausgereizt werden, im Gegensatz zur Workbench sämtliche AmigaDOS-Befehle angewandt werden können. Einerseits ist die Workbench zwar schon so ausgereift, daß die Anwendung der Shell nicht unbedingt nötig ist, es gibt aber immer wieder Fälle, bei denen man nicht um die Kommunikation mit dem Rechner über die Tastatur herkommt. Der Shell und den AmigaDOS-Befehlen ist ein eigener Abschnitt in diesem Buch gewidmet.

2.2.6 Devices (WB 2.1 und 3.0)



In der Schublade Devs stecken auch einige Schmankerln

Neben den Preferences sind die Devices ganz entscheidend mitverantwortlich für die Konfiguration Ihres Rechners. Doch warum steht dann dieses störende (WB 2.1 und 3.0) in der Überschrift? Schließlich haben die Devices schon in den ersten Kickstart-Versionen existiert. Nun das liegt daran, daß wir hier immer noch die Workbench bzw. die zugehörigen System-Disketten betrachten und die Devices erst seit der Workbench-Version 2.1 in die Workbench integriert wurden. So ist es sicher nicht verkehrt, diesen Abschnitt durchzulesen, auch wenn auf früheren Workbench-Versionen nicht alle besprochenen Ordner und auch überhaupt keine Icons existieren.

In der Funktionsweise selbst hat sich seit der ersten Version der Workbench nicht viel geändert, es sind lediglich noch ein paar neue Devices dazugekom-

men. Bisweilen unterscheiden sich die überarbeiteten Devices in ihrer Struktur und in der Erscheinung. Anwender der neuen Workbench 3.0 werden jedoch die meisten der folgenden Dateien vergeblich auf der Diskette suchen, denn wegen Speichermangel wurden nur die notwendigsten Dateien direkt auf der Workbench untergebracht. Der überwiegende Teil jedoch befindet sich auf der "Storage"-Diskette in den jeweils gleichnamigen Ordnern.

2.2.6.1 DosDrivers

Nach dem Öffnen des "Devs"-Ordnern sehen Sie vier weitere Schubladen - bei der Workbench 3.0 hat sogar ein fünftes Verzeichnis Einzug gehalten. Das erste ist mit "DosDrivers" überschrieben und enthält prinzipiell das, was bis einschließlich WB 2.0 in der "Mountlist" steht. Da stellt sich natürlich die Frage, warum man diese Treiber dem Anwender denn so lange vorenthalten hat und wozu sie überhaupt da sind.

Prinzipiell benötigt man Treiber, um auf gewisse Geräte (nichts anderes heißt "Device") zugreifen zu können. So gibt es beispielsweise das "trackdisk.device", das für die Datenübertragung von und zu den Diskettenlaufwerken zuständig ist. Da jeder Amiga mindestens ein Diskettenlaufwerk besitzt, ist dieses "Gerät" kein Thema, es muß immer vorhanden sein.

Schon anders sieht es mit der RAD, der resetfesten RAM-Disk aus. Nicht jeder hat genügend Speicherplatz, um sie betreiben zu können, oder auch gar keinen Bedarf. Andere wiederum benötigen sie ständig. Und genau da setzt das neue Konzept der "modernen" Workbench an. Wenn Sie die RAD benötigen, so brauchen Sie nichts weiter zu tun, als das mit RAD betitelte Icon in den "DOSDrivers"-Ordner zu legen und schon ist sie da! Wollen Sie keine RAD, so werfen Sie das Ding einfach wieder raus, einfacher geht's kaum!

Doch wohin mit dem Ding bzw. noch wichtiger: woher? Genau um diese Frage zu klären, sollten Sie einmal den Ordner namens "Storage", der sich ebenfalls im Hauptverzeichnis befindet, öffnen. Benutzer der Workbench 3.0 werden zu diesem Zweck die Storage-Diskette untersuchen müssen. Hier sind alle vier bzw. fünf Verzeichnisse des "Devs"-Ordnern noch einmal vorhanden. Die stehen da natürlich nicht nur herum, um Ihre Festplatte/Diskette

noch voller zu machen, sondern, wie der Name "Storage" schon ankündigt, sind sie als Aufbewahrungsort für nicht benötigte Devices gedacht. Und aus dem "DosDrivers"-Ordner im "Storage"-Verzeichnis bzw. auf der gleichnamigen Diskette können Sie eben die Treiber holen, die Sie benötigen und die, die Sie nicht brauchen wieder ablegen - für den Fall, daß Sie es sich später einmal anders überlegen.

Diese Vorgehensweise ist bedeutend komfortabler als bei früheren Workbench-Versionen. Früher waren unzählige Arbeitsschritte notwendig, um die gleiche Wirkung zu erzielen, der Umweg über eine Shell oder das CLI war nicht zuletzt ein sehr schwieriges Hindernis für so manchen Amigabesitzer. Doch jetzt wollen wir erst einmal sehen, was uns alles zur Auswahl steht.

Da ist also zunächst die schon angesprochene RAD, die etwa 900 KByte Speicher schluckt, dafür aber ein zu den Diskettenlaufwerken voll kompatibles Speichermedium im RAM erzeugt. Sehr sinnvoll ist Sie beispielsweise beim Installieren von Software von Diskette auf Festplatte. Da kann zuerst mittels Diskcopy die komplette Diskette in die RAD kopiert werden, um die Installation dann aus dem RAM durchzuführen. Dies bringt oftmals einen enormen Geschwindigkeitsgewinn, da das Diskcopy recht schnell Spur für Spur einliest, während bei der Installation selbst meist ständig die gleichen Spuren wieder und wieder gelesen werden müssen.

Dann haben wir den mit "Speak" überschriebenen Treiber zur Sprachausgabe, der eigentlich nicht schaden kann, wenn er im "Devs"-Ordner landet, da er eigentlich kaum Speicher benötigt und - solange er nicht angesprochen wird - auch nicht weiter stört. Stattdessen gehen Sie sicher, daß Programme, die unbedingt diese grausame Art der Kommunikation anwenden wollen, dann auch funktionieren.

Ebenso sollte man nicht auf "Pipe" verzichten, da der Treiber vor allem bei der Shell eine große Bedeutung hat (und wieder einmal: siehe Shell). Doch weil auch von der Workbench gestartete Programme möglicherweise Script-Dateien starten, die auf Pipe zugreifen können, ist es ratsam, ihn immer bereit zu haben.

Weniger interessant ist da schon der "AUX"-Treiber, denn der ist nun wirklich für Sonderanwendungen reserviert. Er behandelt die serielle

Schnittstelle, aber auch nicht in allen Bereichen, sondern nur dann, wenn über den seriellen Port ein Shell-Fenster geöffnet wird. Wenn Sie also nicht gerade ein fortgeschrittener Anwender sind, werden Sie ihn wohl nicht benötigen.

Doch nun - bis zum Schluß aufgespart - kommt der zweite Hammer, den unser neues Betriebssystem zu bieten hat. Denn hinter den unscheinbaren Namen "PC0" und "PC1" befinden sich Treiber zum Lesen und Beschreiben von Disketten im MS-DOS-Format. Wenn Sie also gelegentlich mit von PCs beschriebenen Disketten weiterarbeiten wollen, um z.B. Textfiles zwischen den unterschiedlichen Rechnertypen auszutauschen, so ist nichts weiter zu tun, als die Treiber in den "Devs"-Ordner zu legen. Anschließend können Sie eine MS-DOS formatierte Diskette in ein ganz normales Amiga-Laufwerk einlegen und sie über PC0: bzw. PC1: ansprechen. Dabei ist mit PC0: das Laufwerk, das normalerweise mit DF0: angesprochen wird, gemeint und PC1: bezieht sich dementsprechend auf das Laufwerk DF1:.

Sie können auch unformatierte Disketten auf MS-DOS Format formatieren, und zwar über den ganz normalen "Format"-Befehl. Wenn Sie PC0: oder PC1: angegeben haben, wird in dem entsprechenden Laufwerk eine MS-DOS Diskette erzeugt. Allerdings muß auch erwähnt werden, daß Sie selbstverständlich keine HD-Disketten bearbeiten können, sondern nur das dem 880 KByte auf dem Amiga entsprechende 720 KByte-Format verwenden. Doch die Hauptsache ist ja, daß überhaupt ein Datenaustausch stattfinden kann und noch dazu so einfach.

Das waren also die zur Verfügung stehenden Treiber. Wenn für andere Anwendungen einmal ein Treiber mitgeliefert werden sollte, muß analog vorgegangen werden, d.h. die Treiber einfach in den "DOSDriver"-Ordner des "Devs"-Verzeichnisses legen.

Bitte bedenken Sie auch, daß irgendwelche Veränderungen der Inhalte dieses Ordners erst nach einem Systemneustart (Reset) aktiv werden. Wenn Sie also schnell eine resetfeste RAM-Disk benötigen, ist es der falsche Weg, das Icon kurz in den richtigen Ordner zu ziehen. Vielmehr müssen Sie das entsprechende Icon Doppelklicken, um den Treiber vorübergehend (d.h. bis zum nächsten Systemneustart) zu aktivieren. Dadurch ist es natürlich auch viel einfacher, einen nur sehr selten benötigten Treiber kurzzeitig einzubinden, ohne ihn erst durch die Gegend zu schieben.

2.2.6.2 Keymaps

Bei den "Keymaps" läuft die Sache ein wenig anders. Hier bedeutet die Existenz von Tastaturbelegungen im "Devs"-Ordner nicht, daß diese alle aktiviert werden - es kann ja immer nur eine Einstellung gültig sein. Stattdessen werden alle hier befindlichen Tastaturbelegungen in dem Preferences-Programm "Input" zur Auswahl gestellt. Der "Keymaps"-Ordner im "Storage"-Ordner (bzw. auf der "Storage"-Diskette) hat deshalb eigentlich keine Bedeutung, denn die letzte Entscheidung, welche Tastaturbelegung Sie nun aktivieren, läuft über die Preferences.

2.2.6.3 Monitors

Bei den "Monitors" ist es ebenso. Sie wählen nur die möglichen Monitoreinstellungen, abhängig von Ihrem Monitortyp. Das hat noch überhaupt keine Auswirkungen. Erst im "Screenmode"-Voreinsteller werden abhängig davon mehr oder weniger Bildschirmmodi angezeigt und zur Auswahl angeboten. So sollten Sie als Besitzer eines 1081/1084 sowohl den "PAL"- als auch den "NTSC"-Monitor in den "Monitors"-Ordner des "Devs"-Verzeichnisses ziehen, da dieser Monitortyp beide Einstellungen verkraftet.

Legen Sie allerdings auch noch den "Multiscan" dazu, und besitzen Sie darüber hinaus noch das ECS, so sollten Sie sich davor hüten, die neue Auswahlmöglichkeit bei den "Screenmodes" auszunützen und den "Productivity"-Modus einzustellen. Denn das Programm geht ja davon aus, daß Sie über einen Multiscan-Monitor verfügen. Es würde künftig eine Bildschirmfrequenz verwenden, die Ihr Bildschirm nicht darstellen kann, Sie würden dann überhaupt nichts mehr sehen und die Einstellung dadurch auch nicht mehr rückgängig machen können.

Nicht benötigte Monitortypen können wieder im "Storage"-Verzeichnis bzw. auf der entsprechenden Diskette gelagert werden. Besitzer der Workbench 2.0 haben auch schon Gelegenheit, "Screenmode" mit den nötigen Informationen zu versorgen, und das geht praktisch genauso, wie eben be-

schrieben. Lediglich die Namen der Verzeichnisse unterscheiden sich, hier sind die entsprechenden Ordner direkt im Hauptverzeichnis und nennen sich "Monitors" für die verfügbaren Typen und "MonitorStore" für alle nicht benötigten.

2.2.6.4 - *Printers*

Das bei den "Keymaps" Gesagte gilt analog auch bei der "Printers"-Schublade. Auch hier treffen Sie die Auswahl, welcher Druckertreiber letztlich aktiviert wird, über ein Preferences-Programm, diesmal im "Printer"-Editor. Da bei der Installation aber nur recht allgemeine Druckertypen ausgewählt werden können, bekommen Sie meist mehr Druckertreiber installiert, als Sie letztendlich wollten.

So werden bei der Wahl von HP-Druckern immerhin fünf verschiedene Treiber kopiert, vom HP Deskjet bis zum HP Laserjet. Die Typen, die Sie nicht benötigen, können wieder ins Lager "Storage" wandern - oder bei wirklich akutem Speichermangel auch gelöscht werden (etwa auf der Festplatte, wenn eine Sicherheitskopie vorhanden ist).

2.2.6.5 - *DataTypes (nur bei WB 3.0)*

Der "DataTypes"-Ordner der Workbench 3.0 paßt nicht so recht in das Konzept des DEVS-Verzeichnisses, da die vier darin untergebrachten Dateien weder Treiber noch andere Daten für Voreinstellungen sind. Die vier Dateien dienen einigen Hilfsprogrammen auf der Workbench als Musterdatei, um erkennen zu können, mit welcher Art von Daten sie es bei anderen Dateien zu tun haben.

Es sind Muster von Musikdateien ("8SVX"-Format), Textdateien (nur für Dateien im "FTXT"-Format) und Grafikdateien (im "ILBM"-Format) sowie für Amiga-Hilfsdateien ("AmigaGuide") vorhanden. Bei den ersten drei Dateiformaten handelt es sich um sogenannte IFF-Dateien (IFF steht für Interchange-File-Format), eine Entwicklung der Firma Electronic Arts. Vor allem auf dem Grafik-Sektor konnten sich diese Formate durchsetzen.

2.2.7 - Expansion und WBStartup

Diese beiden Verzeichnisse haben eines gemeinsam: Hier können und dürfen nur bestimmte Dateien oder Programme abgelegt werden, die beim Systemstart bzw. beim Start der Workbench automatisch berücksichtigt werden. Alle anderen Programme oder Dateien haben darin nichts verloren.

2.2.7.1 - WBStartup

Wenn Sie in der Regel nur mit einem Programm arbeiten, ist es sicherlich sinnvoll, wenn dieses Programm automatisch gestartet wird, sobald alle anderen Startvorbereitungen des Amigas (festlegen der Logischen Geräte, öffnen von Shell oder Workbench) abgeschlossen sind. Zu diesem Zweck wurde mit der Workbench-Version 2.0 eine Schublade "WBStartup" eingeführt. Darin können Sie das Piktogramm (Icon) ihres Programmes - und damit das Programm selbst - ablegen, wodurch es bei jedem Laden der Workbench automatisch gestartet wird.

Sehr wirkungsvoll kann beispielsweise dieses Verzeichnis bei den Commodities eingesetzt werden. Wenn Sie dann in den Tooltypes (siehe Information) angeben, daß das Programm kein Fenster öffnen soll, sondern alle für einen sofortigen Einsatz nötigen Parameter schon definieren, wird jedes Hilfsprogramm sofort in den Hintergrund geladen, ohne daß Sie sich weiter darum kümmern müssen.

Sie können den Programmen in den Tooltypes noch drei verschiedene Eigenschaften verleihen:

DONOTWAIT Normalerweise ruft die Workbench erst dann das nächste Programm auf, wenn das vorherige beendet ist. Mit diesem Parameter wartet die Workbench nicht auf das Ende des geladenen Programmes. DONOTWAIT muß immer bei dem Programm angegeben werden, auf das nicht gewartet werden soll.

WAIT=<Zeit> Bis zum Start des nächsten Programmes sollen <Zeit> Sekunden vergehen.

STARTPRI=<Pri> Hier kann man die Priorität angeben, mit der ein Programm gestartet werden soll - also die Reihenfolge festlegen. Standardgemäß haben die Programme die Priorität 0. Das Programm mit der größten Priorität wird als erstes gestartet, das mit der kleinsten ist das letzte, wobei die Werte zwischen -128 und +127 liegen können.

Noch ein kleiner aber nicht unwesentlicher Hinweis: Bedenken Sie immer, daß beim Verschieben von Icons (innerhalb des gleichen Datenträgers) auch das zugehörige Programm nur verschoben und nicht kopiert wird, d.h. es existiert im alten Verzeichnis nicht mehr.

Wollen Sie daher einmal ein Programm aus der WBStartup entfernen, sollten Sie es in das Verzeichnis zurückschieben, aus dem es stammt. Denn wenn Sie zum "Delete" oder "Löschen" greifen, wird das Programm auf diese Weise möglicherweise für immer gelöscht.

2.2.7.2 Expansion

In diesem Verzeichnis haben direkt ausführbare Programme oder andere Daten nichts zu suchen. Dieses Verzeichnis ist dafür vorgesehen, sogenannte Gerätetreiber aufzunehmen. Diese Software dient zum Ansteuern von zusätzlicher Hardware, die an den Amiga angeschlossen wurde. Sollten Sie ein nicht gerade alltägliches Zusatzgerät an Ihren Amiga anschließen wollen, kann es sein, daß eine spezielle Steuerungssoftware benötigt wird. Der mitgelieferte Treiber muß dann in der Expansion-Schublade abgelegt werden, näheres sollte in der jeweiligen Dokumentation zum Gerät erläutert sein.

Bei jedem Neustart des Amigas sorgt der Befehl "Bindddrivers" in der Startup-Sequence dafür, daß dieses Verzeichnis nach zusätzlichen Gerätetreiber durchsucht wird und eventuell gefundene Treiber in das System eingebunden werden.

2.3 - CrossDos (WB 2.1 und 3.0)

Wie im Kapitel zu den Verzeichnisse DEVS/DOSDrivers schon erwähnt wurde, gibt es jetzt auch die Möglichkeit, Disketten im MS-DOS-Format zu verarbeiten. Dabei kommen die beiden Treiber PC0: bzw. PC1: zum Einsatz. CrossDOS ist zum einen der Name eines weiteren Commodities und zum anderen auch der Überbegriff des gesamten Paketes inklusive den Treibern PC0: und PC1:. Wir sollten zunächst einmal die Eigenschaften der Treiber noch genauer untersuchen und anschließend noch das zugehörige Commodity erklären.

2.3.1 - Namenskonventionen unter MS-DOS

Wie beim echten PC dürfen in den Dateinamen natürlich bestimmte Zeichen nicht verwendet werden, da diese bereits anderweitig vergeben sind. Im Einzelnen sind dies folgende Zeichen:

- < > spitze Klammern links und rechts
- . , Punkt, Komma
- “ : doppeltes Anführungszeichen, Doppelpunkt
- | ; Vertikaler Strich, Semikolon
- + = Pluszeichen, Gleichheitszeichen
- / \ Schrägstrich normal und rückwärts
- [] eckige Klammern links und rechts

MS-DOS-Namen bestehen aus maximal acht Zeichen Dateiname, die immer in Großbuchstaben umgesetzt werden und einer Erweiterung mit maximal drei Zeichen, die vom eigentlichen Namen mit einem Punkt getrennt sind.

Da beim Amiga besonders die Dateien mit der “.info”-Endung, die ja bekanntlich für die schönen Bildchen auf der Workbench verantwortlich sind, nicht direkt in die MS-DOS-Norm passen, bekommen diese Dateien von den “PCx:”-Treibern automatisch die Erweiterung “.INF”, umgekehrt werden “.INF”-Dateien im MS-DOS-Format in “.info”-Dateien ins Amiga-Format

umgewandelt. Allerdings darf das zu diesem Icon gehörige Programm selbst keine Namenserweiterung aufweisen, sonst läuft das ganze schief. Das Bildchen der Datei "Lebenslauf.Bruder" (also die Datei "Lebenslauf.Bruder.info") wird in die Datei "LEBENSLA.BRU" konvertiert, bei der Rückkonvertierung ist dann von dem Bildchen weit und breit nichts mehr zu sehen.

Da beim PC die Verzeichnisse im Pfadnamen durch einen sogenannten "Backslash" \ voneinander getrennt werden, beim Amiga jedoch vom normalen "Slash" /, bildet CrossDOS den Kompromiß: es sind beim MS-DOS-Format beide erlaubt.

2.3.2 - CrossDOS - Das Commodity

Zu den neuen Treibern zum Ansprechen der Laufwerke im PC-Format (PC0: und PC1:) gehört auch ein Commodity. Da der ASCII keineswegs ein echter "Standardcode" ist, wie er es von sich behauptet, sondern immer noch viel Platz für eigene Definitionen läßt, hat natürlich auch jede Firma ihre eigene Variante. Das fällt im normalen Betrieb nicht weiter auf, da die ersten Codes alle noch wohldefiniert sind, doch bereits bei Umlauten merkt man eben doch, daß die Definitionen nicht vollständig sind. Daher ist ein "ä" auf dem Amiga kein "ä" auf dem PC und umgekehrt. Dazu kommt dann noch die Eigenheit von PCs, nicht nur ein Linefeed ans Zeilenende anzuhängen, wie es beim Amiga der Fall ist, sondern auch noch ein Carriage Return. Deshalb sehen Texte, die man 1:1 vom PC übernommen hat, immer etwas seltsam aus.

Um eben diesen Effekt zu umgehen, kann man Textfilter und Textkonvertierung einschalten, die sich selbstverständlich wirklich nur auf Texte beziehen, Programme können Sie auf einem normalen Amiga ohnehin nicht starten. Außerdem läßt sich noch die Konvertierungsart bestimmen und das ganze jeweils getrennt für jedes einzelne Laufwerk.

Mit dem Gadget "Textfilter" werden die Linefeeds an jedem Zeilenende je nach Konvertierungsrichtung angefügt oder gelöscht, die am Dateiende vorhandenen "EOF"-Zeichen ("EOF" steht für "End of File") werden gegebenenfalls ergänzt bzw. entfernt.

Mit "Textkonvertierung" können Sie die Umlautanpassung ein- oder ausschalten und die Konvertierungsart gibt an, welche Zeichen wie umgewandelt werden sollen. Bei "ASCII-7" wird der normalerweise in 8-Bit-Zeichen vorliegende Text in das 7-Bit-Zeichen-Standardformat konvertiert. Mit "INTL" wird versucht, die länderspezifischen Umlaute bei der Konvertierung beizubehalten, so daß sie später wieder verfügbar sind, mit der Auswahl von "Dansk" können Sie speziell dänische Texte und deren Besonderheiten behandeln und konvertieren.

Noch ein Hinweis zum Schluß: Da das CrossDOS-Commodity nicht erkennen kann, ob die zu konvertierende Datei eine Textdatei ist oder nicht, wird jede Datei bei eingeschalteten Filter durchforstet und entsprechend geändert. Wenn es sich dabei nicht um eine Text- sondern etwa um eine Grafik-Datei handelt, werden dadurch Informationen verfälscht - oder schlimmstenfalls zerstört.

Sie sollten daher dieses Commodity dann und nur dann verwenden, wenn Sie augenblicklich einen Text auf einer MS-DOS-Diskette speichern wollen, oder davon laden wollen. Ansonsten sollte die Textkonvertierung stets inaktiv sein.

3 - Amiga DOS

3.1 - Grundlegendes zu AmigaDOS

Der Begriff "AmigaDOS" ("DOS" steht für "Disk Operating System") umfaßt eigentlich weit mehr, als die Verwendung im alltäglichen Sprachgebrauch. AmigaDOS ist im wesentlichen ein sehr wichtiger Teil des Betriebssystems des Amiga.

Es ist vor allem verantwortlich für den Datenaustausch zwischen den Programmen und den angeschlossenen Geräten (Diskettenlaufwerk, Drucker, Modem) und der Verwaltung der Daten auf Speichermedien wie Diskette oder Festplatte. Die Aufgabe von AmigaDOS ist, dem Anwender (und auch dem Programmierer) umfangreiche Hilfsmittel zur Daten- und Dateiverwaltung bereitzustellen. So kann der Programmierer auf eine Sammlung von Systemroutinen, die sogenannte "dos.library" zurückgreifen, für den Anwender existiert eine große Zahl von Befehlen beispielsweise zum Formatieren von Disketten oder kopieren von Dateien.

Wenn jedoch jemand vom AmigaDOS spricht, so meint er in der Regel nur die Befehle und die Benutzeroberfläche "Shell" (früher als "CLI" für "Command Line Interface" genannt), die vom Betriebssystem für den Anwender bereitgestellt werden. Dafür ist nicht zuletzt auch die Literatur über den Amiga verantwortlich, denn meist werden die Benutzeroberfläche und die Befehle mit dem "AmigaDOS" gleichgesetzt, ohne auf die eigentlich umfassendere Bedeutung hinzuweisen.

Auch wir können hier nur den anwenderbezogenen Teil behandeln, und überlassen die für Programmierer relevanten Informationen anderer Fachliteratur.

3.2 - Shell - die Benutzeroberfläche

Die Befehle, die das AmigaDOS zur Verfügung stellt, müssen ja dem Computer mitgeteilt werden. Dies geschieht über die Benutzeroberfläche

“Shell”, die eine reine Textschnittstelle darstellt, d.h. die Befehle müssen über die Tastatur eingegeben werden.

Dafür wird ein Fenster (häufig auch Konsolenfenster genannt) geöffnet, das den Workbench-Fenstern bis auf die fehlenden Rollbalken gleicht. Aber es gibt noch weitere Unterschiede zu den Workbench-Fenstern. So können keine Icons in das Fenster gezogen werden und die Hintergrundmuster der Workbench erscheinen in diesem Fenster nicht.

Mit dem Programm Fonts aus dem Prefs-Directory können Sie für dieses Fenster aber einen eigenen Zeichensatz festlegen. Sie sollten dabei jedoch beachten, daß Sie hierfür einen Font auswählen, dessen Zeichen einen konstanten Abstand voneinander haben und es sich dabei nicht um einen sogenannten proportionalen Zeichensatz handelt. Andernfalls ist es für den Cursor der Shell schwierig, schon eingegebene Zeilen bzw. deren Zeichen nachträglich zu verändern, was andererseits sicherlich sehr häufig notwendig ist.

Jetzt wurden die Eigenschaften schon so ausführlich beschrieben, ohne daß ein Wort darüber verloren wurde, wie man denn überhaupt ein Shell-Fenster öffnen kann. Hierzu gibt es zwei Möglichkeiten, zum einen kann man das entsprechende Icon auf der Workbench im Verzeichnis System anwählen. Dadurch wird das eigentliche Programm gestartet, daß interessanterweise nicht “Shell” sondern “CLI” (wie einst bei den ersten Versionen) heißt. Zum anderen können Sie in einer schon vorhandenen Shell mit den Befehlen “CLI”, “NewCLI” oder “Newshell” weitere Fenster öffnen, die dann alle voneinander unabhängig sind. So können in verschiedenen Shell-Fenstern gleichzeitig verschiedene Befehle abgearbeitet werden.

3.2.1 - Editieren in der Shell

In einem Shell-Fenster können Sie immer nur eine Zeile eingeben, oder verändern. Dabei stellt die Shell viele Hilfsmittel in Form von verschiedenen Tastenkombinationen zur Verfügung, die die Eingabe von Befehlen wesentlich erleichtern und den Anwender unterstützen.

Die Funktionen im Einzelnen:

Zeilenmanipulation:

Cursor links	rückt den Cursor ein Zeichen nach links
Cursor rechts	rückt den Cursor ein Zeichen nach rechts
Shift-Cursor links	setzt den Cursor an den Zeilenanfang
Shift-Cursor rechts	setzt den Cursor an das Zeilenende
Backspace	löscht das Zeichen links vom Cursor
Del	löscht das Zeichen unter dem Cursor
Ctrl-H	löscht das Zeichen links vom Cursor (wie Backspace)
Ctrl-M	führt Befehlszeile aus (wie Return-Taste)
Ctrl-J	Zeilenvorschub ohne Befehlsausführung
Ctrl-W	löscht vom Cursor bis zum nächsten linken Leerzeichen
Ctrl-X	löscht die ganze Zeile
Ctrl-K	löscht den gesamten Text vom Cursor bis zum Zeilenende
Ctrl-Y	holt die mit Ctrl-K gelöschten Zeichen wieder
Ctrl-U	löscht die Zeichen vom Zeilenanfang bis zur Cursorposition

Befehlsvorgeschichte:

Cursor aufwärts	Früher eingegebene Zeilen holen
Cursor abwärts	später eingegebene Zeilen holen
Shift-Cursor aufwärts	zeigt die letzte Zeile an, deren Anfang mit der aktuellen übereinstimmt
Shift-Cursor abwärts	springt an das Ende der Befehlsvorgeschichte
Ctrl-R	wie "Shift-Cursor aufwärts"

Prozeßsteuerung:

beliebige Taste	stoppt die Bildschirmausgabe des laufenden Befehles, sofern diese Taste ein druckfähiges Zeichen war.
Backspace	setzt die Ausgabe fort (ebenso Return und Ctrl-M)
Ctrl-C	die Bearbeitung des Befehls wird abgebrochen
Ctrl-D	Wird ein Befehls-Script ausgeführt, wird dieses abgebrochen
Ctrl-S	Ausgabe wird angehalten
Ctrl-Q	Ausgabe wird fortgesetzt, wenn sie mit Ctrl-S angehalten wurde
Ctrl-\	Stellt die normale Befehlszeileneingabe wieder her oder schließt das Fenster, wenn die normale Eingabe hergestellt ist.

Hier sind jetzt einige neue Begriffe aufgetaucht, die einer näheren Betrachtung bedürfen. Doch vorher muß ich noch eine Kleinigkeit richtig stellen.

Ich habe oben erwähnt, daß es in einem einzigen Shell-Fenster nicht möglich ist, mehr als nur eine Zeile - und damit nur einen Befehl - einzugeben. Es ist jedoch möglich, mehrere Befehle einzugeben, bevor die Bearbeitung gestartet wird. Wird nach jedem Befehl die Tastenkombination Ctrl-J gedrückt, so springt der Cursor in eine neue Zeile, ohne daß die vorherige Zeile ausgeführt wird. Allerdings interpretiert AmigaDOS die einzelnen Zeilen als eine einzige. Es kann daher wirklich nur immer eine Zeile eingegeben werden - aus Sicht des Amigas.

Nun aber noch ein paar Sätze zur obigen Tabelle.

Zu den Möglichkeiten zur Zeilenmanipulation kann nicht viel gesagt werden, wohl aber zur Befehlsvorgeschichte. Das AmigaDOS speichert alle Zeilen, die in einem Fenster eingegeben werden in einem 2 KB großen Puffer. Dabei hängt es von der Länge der einzelnen Zeilen ab, wieviele letztenendes gespeichert

chert werden können. Wenn eine neue Zeile nicht mehr in den Puffer paßt, wird damit begonnen, die ersten Zeilen wieder zu löschen. Die einzelnen Zeilen in diesem Puffer können mit den Tasten "Cursor aufwärts" und "Cursor abwärts" wieder hergeholt werden. So können Sie bequem einen Befehl beliebig oft wiederholen, in dem Sie ihn einmal eingeben und dann immer "Cursor aufwärts" und Return drücken. Dabei ist zu beachten, daß Befehle, die auf diese Weise beliebig oft wiederholt werden, im Puffer nur einmal erscheinen. Wenn Sie eine Befehlszeile brauchen, die Sie schon einmal etwas früher eingegeben hatten (aber immer nur in einem Fenster - der Puffer ist an eine Shell gebunden), so können Sie diese Zeile direkt holen, indem Sie den Beginn dieser Zeile als Muster wieder eingeben, und die Shell mit der Kombination "Shift-Cursor aufwärts" nach einer Zeile, die zu diesem Muster paßt, suchen lassen. Wird eine passende Zeile gefunden, können Sie den Befehl einfach wieder ausführen lassen, indem Sie die Return-Taste drücken. Hierbei wird diese Zeile erneut in den Befehls-Puffer aufgenommen.

Das AmigaDOS kann jedoch mitunter Probleme bereiten, wenn Sie mit den Cursortasten durch den Befehls-Puffer gehen und dieser schon gefüllt ist. In diesem Fall wird es manchmal schwierig, die Leerzeile - also das Ende des Puffers - zu finden, sogar die Tastenkombination "Shift-Cursor abwärts" kommt mit dieser Situation nicht zurecht. Es gibt aber ein paar Tricks, um schnell wieder an das Ende zu gelangen. Man kann eine beliebige Zeile verwenden, als erstes Zeichen ein Semikolon ";" einfügen und Return drücken. Durch das Semikolon interpretiert AmigaDOS die ganze Zeile als Kommentar und führt keinen Befehl aus. Anschließend wird eine Leerzeile angezeigt, man ist wieder am Pufferende. Und wer sich die Kombination Ctrl-X merken kann, der hat solche Umwege gar nicht nötig.

Seit der Workbench Version 3.0 bietet AmigaDOS eine weitere interessante Möglichkeit an. Sie können jetzt Texte von einem Fenster in ein anderes übertragen, ohne sämtliche Zeilen neu abzutippen. Dazu markieren Sie den Text, den Sie kopieren wollen, mit der Maus. Dies funktioniert analog zur Mehrfachauswahl von Icons mit dem Gummiband auf der Workbench. Anschließend drücken Sie die rechte Amigataste und "C". Nun können Sie diesen so markierten Text an anderer Stelle wieder einfügen, indem Sie an gewünschter Stelle wieder die linke Maustaste drücken. Mit der rechten Amiga-Taste in Verbindung mit "V" wird der markierte Text eingefügt.

3.2.2 - Die Eigenschaften des Shell-Fensters

Der Anwender muß die Eigenschaften eines Shell-Fensters nicht als gegeben hinnehmen, er kann verschiedenes sogar selbst bestimmen. So können Sie mit dem Tool Type "WINDOW" (siehe Menüpunkt "Information" des Workbench-Menüs) beispielsweise die Größe, Position, Fenstertitel und vieles andere den eigenen Ansprüchen anpassen. Das WINDOW-Merkmal hat dabei folendes Format:

WINDOW=CON:x/y/Breite/Höhe/Titel/Option

Die Bedeutung der einzelnen Parameter:

x,y	Abstand der linken oberen Ecke des Fensters vom linken oberen Eck des Bildschirms in Pixel (Bildschirmpunkten)
Breite	Fensterbreite in Pixel
Höhe	Fensterhöhe in Pixel
Titel	Text im Titelbalken des Fensters

Mögliche Optionen:

CLOSE	Das Fenster erhält neben den normalen Gadgets auch das Schließsymbol.
AUTO	Das Fenster öffnet sich automatisch, wenn eine Eingabe erwartet wird oder ein Programm Informationen ausgibt. Handelt es sich um ein echtes Shell-Fenster, öffnet es sich sofort und kann ausschließlich mit dem ENDSHELL-Befehl wieder geschlossen werden. Bei Auswahl des Schließsymbols wird das Fenster zwar geschlossen, aber gleich wieder geöffnet, weil es wieder eine Eingabe erwartet.

BACKDROP	Geben Sie einem Fenster diese Eigenschaft, wird es hinter allen anderen Workbenchfenstern geöffnet. Es enthält nur das Zoom-Gadget und kann nicht in den Vordergrund geholt werden. Wenn Sie dieses Fenster benutzen wollen, müssen Sie alle anderen Fenster entweder schließen oder soweit verkleinern, daß es sichtbar wird.
NOBORDER	Bei diesem Fenster fehlen der linke und obere Rahmen. Als Gadgets stehen "Zoom", "Vorder/Hintergrund" und "Größe" zur Verfügung.
NODRAG	Dieses Fenster kann nicht verschoben werden, es enthält alle Gadgets außer dem Schließsymbol.
NOSIZE	Hier gibt es nur das "Vorder/Hintergrund"-Gadget.
SCREEN	Hier kann angegeben werden, auf welchen Screen das Fenster geöffnet werden soll. Hinter dem Schlüsselwort "SCREEN" muß der Name des Bildschirms angegeben werden, dieser Bildschirm muß bereits geöffnet sein.
SIMPLE	Diese Eigenschaft ist voreingestellt und braucht normalerweise nicht angegeben werden. Diese Option steht für die Eigenschaft, daß AmigaDOS bei einer Änderung der Fenstergröße den angezeigten Text anpaßt und gegebenenfalls auch den Text, der schon aus dem Fenster wieder verschwand, bei einer Vergrößerung wieder angezeigt wird.
SMART	Hier wird die Anzeige bei einer Veränderung der Fenstergröße nicht angepaßt.
WAIT	Dieses Fenster kann nur durch Anklicken des Schließ-Symboles wieder geschlossen werden.

Diese Optionen lassen sich beliebig miteinander kombinieren. Dabei werden die Schlüsselwörter der einzelnen Optionen nur durch Leerzeichen voneinander getrennt. Sie sollten aber dennoch bei der Auswahl der einzelnen Optionen Vorsicht walten lassen, da es auch ungünstige Kombinationen gibt, so sollten Sie beispielsweise die Option "WAIT" nie ohne die Option "CLOSE" verwenden.

Einige der oben erwähnten Optionen scheinen auf den ersten Blick sinnlos zu sein, da eine Anwendung in einem normalen Shell-Fenster undenkbar ist. Es gibt aber auch andere Möglichkeiten, derartige Fenster zu öffnen, bei denen die eine oder andere Option durchaus notwendig wird. Denken Sie beispielsweise an den Menüpunkt *Execute Command* der Workbench. Es wäre doch sehr ungünstig, wenn dieses Fenster zwar mit der *AUTO*-Option aber nicht mit der *WAIT*-Option ausgestattet wäre.

Jedesmal wenn eine neue Shell geöffnet wird, wird die Datei *s:Shell-startup* - falls vorhanden - aufgerufen. In diese Datei kann der Anwender seinen individuellen Wünschen entsprechend die Shell-Umgebung anpassen, etwa die Eingabeaufforderung ändern (mit dem Befehl *PROMPT*), zusätzliche Suchpfade bestimmen (über den Befehl *PATH*) oder für bestimmte Befehle Abkürzungen schaffen (mittels *ALIAS*). Diese Befehle werden im folgenden Kapitel ausführlich beschrieben.

3.2.3 - Die Escape-Sequenzen nach ANSI-X3.64

Es ist möglich, im Konsolenfenster die Textdarstellung in der Farbe oder Schriftart (fett, kursiv, unterstrichen oder Kombinationen davon) zu verändern oder auch neue Randeinstellungen vorzunehmen. Dazu verwendet AmigaDOS sogenannte Escape-Sequenzen, das sind Steuerzeichenfolgen, die mit der Escape-Taste oder dem entsprechendem ASCII-Code 27 (in hexadezimal 1B) eingeleitet werden und im Standart ANSI-X3.64 festgelegt sind. Auf das Escape-Zeichen können ein oder mehrere Zeichen oder auch Zahlen folgen, Leerzeichen werden in der Regel nicht verwendet.

Die folgende Tabelle enthält die wichtigsten Escape-Sequenzen für den Amiga, die im Format "ESC[#X" (bei den Drucker-Sequenzen allgemeiner "ESC...") angegeben sind. Drücken Sie in der Shell die Esc-Taste, so erscheint eine invertierte linke eckige Klammer, an der sich eine normale linke eckige Klammer [anschließt. Es folgt eine Zahl und ein Buchstabe. Die Escape-Sequenzen unterscheiden zwischen Groß- und Kleinbuchstaben, d.h. ein großer Buchstabe hat eine andere Funktion als ein kleiner. Sollten Sie auf Ihrer Tastatur keine eckigen Klammern finden, drücken Sie eine Alt-Taste und eine der beiden links neben der Return-Taste in der oberen Reihe.

Escape-Sequenzen für das Konsolen-Fenster:

ESCc	macht alle vorhergehenden Änderungen rückgängig
ESC[0m	setzt die Grafikmodi auf die Voreinstellung zurück
ESC[1m	Fettdruck
ESC[3m	Kursivdruck
ESC[4m	Unterstreichen
ESC[7m	invertierte Schrift
ESC[8m	Text in Hintergrundfarbe darstellen
ESC[22m	macht Fettschrift-Umstellung rückgängig
ESC[23m	macht Kursivschrift-Umstellung rückgängig
ESC[24m	macht Unterstreichen rückgängig
ESC[27m	macht invertierte Schrift rückgängig
ESC[28m	stellt normale Textfarbe wieder her
ESC[30m	schreibt Text in Farbe 0 (Hintergrundfarbe)
ESC[31m	schreibt Text in Farbe 1 (voreingestellte Textfarbe)
ESC[3#m	schreibt Text in Farbe # (2-8)
ESC[39m	schreibt Text wieder in der Vorgabe-Farbe (Farbe 1)
ESC[4#m	setzt als Texthintergrund die Farbe # (0-8)
ESC[49m	setzt als Texthintergrund die voreingestellte Farbe 0
ESC[#u	maximale Zeilenlänge auf # Zeichen setzen
ESC[#t	maximale Anzahl Zeilen im Fenster auf # Zeilen setzen
ESC[#x	Text beginnt im Abstand von # Pixeln vom linken Fensterrand
ESC[#y	Text beginnt im Abstand von # Pixeln vom oberen Fensterrand

Die Escape-Sequenzen für die Konsolenfenster werden ausgeführt, sobald die Return-Taste gedrückt wird oder eine Zeichenfolge mit der entsprechenden Sequenz ausgegeben wird. Da es aber schwierig ist, das Esc-Zeichen in Zeichenketten einzubauen weil es im Grunde ein nichtdruckbares Zeichen ist, wurde hierfür eine Alternative geschaffen. So kann das Esc-Zeichen durch die Zeichenfolge *E ersetzt werden. Die Escape-Sequenzen gelten immer nur für das Fenster, in dem es direkt über die Tastatur eingegeben wurde oder in dem es über eine Zeichenkette (etwa von einem Programm) ausgegeben wurde.

Es existieren aber nicht nur für die Fenster Escape-Sequenzen, auch um die Druckeransteuerung zu vereinheitlichen, existieren standardisierte Befehle. Die unten aufgelisteten Sequenzen können eingesetzt werden, wenn ein angeschlossener Drucker als logisches Gerät "prt:" angesprochen wird, denn hier sorgt der im Programm Printer der Preferences-Schublade eingestellte Druckertreiber dafür, daß die einzelnen Escape-Sequenzen in für den Drucker verständliche Befehle umgewandelt werden. Näheres zum Gerät prt: erfahren Sie auch im Kapitel über logische Geräte.

Escape-Sequenzen für Drucker-Ansteuerung:

ESCc	Drucker-Reset
ESC#1	Drucker initialisieren (hier muß das Doppelkreuz # tatsächlich eingesetzt werden).
ESCD	Zeilenvorschub
ESCE	Return-Zeichen
ESCM	Return-Zeichen rückwärts
ESC[0w	setzt normale Zeichengröße
ESC[2w	schaltet Zeichengröße "Elite" ein
ESC[1w	schaltet Zeichengröße "Elite" aus
ESC[4w	schaltet Zeichengröße "Fine" ein
ESC[3w	schaltet Zeichengröße "Fine" aus
ESC[6w	verdoppelt Zeichengröße

ESC[5w	Zeichengröße wieder normalisieren (nach dem Verdoppeln)
ESC[6"z	schaltet Schattendruck ein
ESC[5"z	schaltet Schattendruck aus
ESC[4"z	schaltet Doppeldruck ein
ESC[3"z	schaltet Doppeldruck aus
ESC[2"z	schaltet NLQ ein
ESC[1"z	schaltet NLQ aus
ESC[2v	schaltet Hochstellen ein
ESC[1v	schaltet Hochstellen aus
ESC[4v	schaltet Tiefstellen ein
ESC[3v	schaltet Tiefstellen aus
ESC[0v	macht alle ESC[#v-Befehle rückgängig
ESC[2p	schaltet Proportionalschrift ein
ESC[1p	schaltet Proportionalschrift aus
ESC[0p	löscht Proportionalschrift
ESC[# E	setzt Zeichenabstand auf # Punkte
ESC[5 F	Text linksbündig setzen
ESC[7 F	Text rechtsbündig setzen
ESC[6 F	Text links- und rechtsbündig setzen (Blocksatz)
ESC[1 F	Text zentrieren
ESC[0 F	macht die letzten vier Befehle rückgängig
ESC[0z	setzt Zeilenabstand auf 1/8 Zoll (8 Zeilen je Zoll)
ESC[1z	setzt Zeilenabstand auf 1/6 Zoll (6 Zeilen je Zoll)
ESC[#t	Papierlänge #
ESC[#q	Seitenvorschub #

ESC[0q	kein Seitenvorschub
ESC[P#1;P#2r	setzt oberen Rand auf #1 und unteren Rand auf #2
ESC[P#1;P#2s	setzt linken Rand auf #1 und rechten Rand auf #2
ESC#3	löscht alle Randmarken
ESCH	setzt horizontalen Tabulator
ESCJ	setzt vertikalen Tabulator
ESC[0g	löscht diesen horizontalen Tabulator
ESC[3g	löscht alle horizontalen Tabulatoren
ESC[1g	löscht diesen vertikalen Tabulator
ESC[4g	löscht alle vertikalen Tabulatoren
ESC#4	löscht alle vertikalen und horizontalen Tabulatoren
ESC#5	setzt die voreingestellten Tabulatoren
ESC[P#’x	leitet erweiterte (druckerspezifische) Befehle ein

Für den Drucker können auch einige Escape-Sequenzen aus der Tabelle für die Fenster verwendet werden, wie etwa die Umstellungen auf Kursivschrift oder unterstreichen. Es gibt auch unbrauchbare Befehle, etwa die Festlegung der Startposition des Textes in Pixel vom Fensterrand - ein Drucker kennt keine Fenster.

3.2.4 - Programmaufruf in der Shell

Bei den früheren Amigas war die Tatsache, daß von der Shell wirklich alle Programme gestartet werden konnten - von der Workbench jedoch nicht - der wesentliche Grund dafür, daß oft mit der Shell gearbeitet wurde. Beispielsweise benötigte ich für meine Arbeit am guten alten Amiga 500 wirklich nur in Ausnahmefällen die Workbench. Bei den modernen Amigas existiert dieses Privileg der Shell nicht mehr, da - zumindest theoretisch - sämtliche Programme jetzt auch über die Workbench aufgerufen werden können, dazu müssen sie nicht einmal mehr mit einem Bildchen versehen wer-

den. Dennoch gibt es auch heute noch verschiedene Situationen, in denen die Shell ein komfortableres Arbeiten ermöglicht, als die graphische Benutzeroberfläche. Um Programme von der Shell aus zu starten, muß der Programmname hinter der Eingabeaufforderung (Prompt) eingegeben und die Return-Taste gedrückt werden. Auch die meisten AmigaDOS-Befehle sind nichts anderes als kleine Programme, die jedoch häufig noch zusätzliche Angaben (man nennt diese Angaben Argumente) benötigen, um richtig funktionieren zu können. Benötigt ein Programm noch zusätzliche Argumente, werden diese in der Regel in einer vorgeschriebenen Reihenfolge hinter dem Programmnamen mit Leerzeichen voneinander getrennt eingegeben, und erst dann die Return-Taste gedrückt.

Beispiele:

1> MEmacs

Hier wird das Programm MEmacs geladen und gestartet.

1> MEmacs Text

Hier wird das Programm MEmacs gestartet und die Datei "Text" automatisch geladen.

Die Möglichkeit, zusätzliche Argumente übergeben zu können, erleichtert in erster Linie die Arbeit, da andernfalls diese Informationen über verschiedene Gadget- und/oder Menüaufrufe eingegeben werden müßten. Es gibt aber Programme, die viele Möglichkeiten nur über Argumente beim Programmstart anbieten, nach dem Programmstart können diese dann nicht mehr ausgenutzt werden.

Es kann schon ab und zu einmal vorkommen, daß man nicht mehr genau weiß, welche Argumente ein Befehl benötigt bzw. unterstützt. In diesem Fall kann bei allen AmigaDOS-Befehlen (und auch bei vielen anderen Programmen) durch die Angabe eines Fragezeichens hinter dem Befehlsnamen die Hilfsfunktion aktiviert werden. Hier werden dann alle möglichen Argumente in Form einer Schablone dargestellt. Der Cursor wartet am Ende der Zeile auf die Eingabe der Argumente, so daß der Befehl noch immer ausgeführt werden kann.

Wurde ein Programm von der Shell gestartet, so läuft es anstelle der Shell, d.h. die Shell kann für diese Zeit keine weiteren Eingaben verarbeiten - auch

wenn die Eingabe in das Shell-Fenster möglich ist. Da der Amiga jedoch ein Multitasking-Rechner ist, also gleichzeitig mehrere Programme laufen können, gibt es die Möglichkeit, durch den Befehl RUN ein Programm von der Shell abzulösen und als eigenständiger Prozeß laufen zu lassen. Mit der Shell kann anschließend weiter gearbeitet werden, während das Programm läuft.

Beispiel:

RUN MEMacs

Hier wird MEMacs als eigenständiger Prozeß gestartet, die Shell wird dadurch nicht blockiert. Es erscheint wieder die Eingabeaufforderung (Prompt) - sofern das Programm nicht das Shell-Fenster überlagert. Ist dies jedoch der Fall, können Sie in der Regel mit den Vor-/Hintergrund-Gadgets das Shellfenster nach vorne holen und weiterarbeiten. Etwas gewöhnungsbedürftig ist, daß alle Ausgaben auch des mit RUN gestarteten Programmes in das Shell-Fenster erfolgen, von dem aus es gestartet wurde. So kann es passieren, daß Sie einen weiteren Befehl eingeben wollen und das Programm gerade in diesem Augenblick eine Meldung ausgibt und damit Ihre Eingabe stört. Ist dies der Fall, sollten Sie das Programm nicht mit RUN starten, sondern eine weitere Shell öffnen, in der dann der RUN-Befehl überflüssig ist. Aus diesem Grund kann auch ein Shell-Fenster nicht geschlossen werden, solange noch Programme aktiv sind, die von diesem Fenster aus mit RUN gestartet wurden. Wenn Sie aber auf Meldungen von dem gestarteten Programm keinen Wert legen, existiert noch die Möglichkeit, die Ausgabe in das logische Gerät NIL: (also ins Nichts) umzuleiten. Dann werden Sie zum einen nicht weiter gestört und zum anderen kann das Fenster jederzeit geschlossen werden.

Beispiel:

I> RUN MEMacs >NIL: Text

Hier wird das Programm MEMacs als eigenständiger Prozeß gestartet, sämtliche Meldungen werden ins Nichts umgeleitet. Das Programm lädt automatisch die Datei "Text". Anschließend kann das Shell-Fenster geschlossen werden, ohne daß das Programm MEMacs beendet ist.

Mehr hierzu lesen Sie bei den Befehlen RUN und <,>, sowie im Kapitel über die logischen Geräte.

3.3 - Das Verzeichnissystem der Datenträger

Die Struktur von Datenträgern ist auf der Workbench sehr deutlich zu sehen. Hier klicken Sie zuerst einmal das Symbol eines Datenträgers - in der Regel einer Diskette - an, es erscheint ein Fenster, in der zum einen Dateien und zum anderen meist weitere Schubladen zu finden sind. Sie können daraufhin weitere Schubladen öffnen, in denen wieder Schuladen und Dateien vorhanden sind. Diese sehr anschauliche Struktur entspricht auch dem wirklichen Aufbau des Datenträgers, wenn man von der Möglichkeit absieht, das seit kurzem auch Bildchen ausgelagert werden können (mit dem Menüpunkt "Leave out") und daher die Dateien nicht unbedingt dort stehen müssen, wo deren Icons zu sehen sind.

Zunächst einmal sind AmigaDOS-Disketten (und auch Festplatten) immer durch den Namen des Datenträgers gekennzeichnet. Um nun die vorhandenen Dateien ordnen zu können, gibt es eine zusätzliche Unterteilung in verschiedene Verzeichnisse, die wiederum Unterverzeichnisse enthalten können, bis am Ende die eigentliche Datei steht. Diese Verzeichnisse werden auf der Workbench als Schubladen dargestellt. Auch in der Shell muß diese Struktur irgendwie ausgedrückt werden. Dazu wurden sogenannte Pfade oder Pfadnamen geschaffen, die allgemein wie folgt aussehen:

Name: *Verzeichnis1/Verzeichnis2/.../Datei*

Man beschreitet auf seinem Weg praktisch einen Pfad durch die Verzeichnisstruktur, und daher kommt dann auch der Begriff "Pfadname". Anstelle des Namens eines Datenträgers kann auch der Gerätenamen des Laufwerkes verwendet werden, in dem der Datenträger ist ("df0:", "hd1:" usw.). In der Shell sieht der komplette Pfad dann beispielsweise folgendermaßen aus:

Workbench: *Devs/Keymaps/d*

Der Name des Datenträgers wird also mit einem Doppelpunkt, die Verzeichnisse mit Schrägstrichen abgegrenzt. Was Sie jedoch auf der Workbench davon sehen, sind nur die Schubladen und Fenster, die Sie nach

und nach öffnen. Liegt die Diskette "Workbench" beispielsweise im Laufwerk "df1:", kann die Datei "d" auch so angesprochen werden:

df1:Devs/Keymaps/d

Von diesem System der Strukturierung von Datenträgern sollte auch ausgiebig Gebrauch gemacht werden, da sonst spätestens bei der Benutzung von Festplatten die Übersicht verloren geht. Der Amiga setzt theoretisch keine Grenzen bei den Verzeichnistiefen, eine physikalische Grenze ist jedoch vorhanden, da jeder Speicher gewisse Grenzen aufweist. Sollte irgend ein Programm jedoch mit zunehmender Verzeichnistiefe Probleme bereiten, könnte eine Vergrößerung des sogenannten Stacks (Stapelspeicher) möglicherweise Abhilfe schaffen. Dazu muß mit dem STACK-Befehl (siehe auch dort) der Stapelspeicher vergrößert werden.

Jeder Name eines Verzeichnisses oder einer Datei kann bis zu 30 Zeichen lang sein, wobei neben Groß- und Kleinbuchstaben auch Satzzeichen ohne Sonderfunktion verwendet werden dürfen. Nicht erlaubt sind dagegen: Doppelpunkt (:), Semikolon (;), Stern (*), Schrägstrich (/), umgekehrter Apostroph (') und das Prozentzeichen (%). Die Groß- und Kleinschreibung wird bei der Benennung von Verzeichnissen oder Dateien beibehalten, obwohl das AmigaDOS selbst die Groß- und Kleinbuchstaben nicht unterscheidet, also letztenendes nur die Zeichenfolge ausschlaggebend ist. So wird das Verzeichnis "HotelVerw" sowohl durch "hotelverw", "HOTELVERW" als auch "HotElVERW" angesprochen.

Von der Verwendung von Leerzeichen in Dateinamen kann nur abgeraten werden, zur Trennung von Wörtern sollte es durch das Unterstreichungszeichen _ ersetzt werden, damit der Dateiname auch in der Shell als zusammenhängend erkannt wird.

ACHTUNG: Am Anfang oder Ende eines Namens soll unter keinen Umständen ein Leerzeichen verwendet werden, da diese weder in der Workbench noch in der Shell erkennbar sind. Diese müßten jedoch immer eingegeben werden, damit AmigaDOS den eingegebenen Namen der Datei zuordnen kann.

Es ist möglich, zwei verschiedenen Dateien den gleichen Namen zu geben, sofern sie nicht im selben Verzeichnis stehen.

Es kann sein, daß Datei- oder Verzeichnisnamen mit Schlüsselwörtern (bestimmte Argumente bei den einzelnen Befehlen) übereinstimmen. Um dennoch die Namen unterscheiden zu können, sollten diese bei den betreffenden Befehlen in Anführungszeichen gesetzt werden.

Beispiel: Es existiert das Unterverzeichnis "Files" im aktuellen Verzeichnis. Wenn Sie den Inhalt dieses Verzeichnisses sehen wollen, müssen Sie den Befehl

dir "Files"

eingeben. Wird dagegen

dir Files

einggegeben, so werden nur die Dateien im aktuellen Verzeichnis angezeigt.

Die Zeichen, die nicht in Verzeichnis- oder Dateinamen verwendet werden dürfen, sind schon fest verplant. So zeigt der Doppelpunkt am Ende einer Zeichenkette immer, daß es sich hier um ein logisches oder physikalisches Gerät handelt (beispielsweise "df0:", "prt:", "c:" oder "Workbench:"). Schrägstriche (/) werden ausschließlich zur Trennung von Verzeichnis- und Unterverzeichnisnamen bzw. Verzeichnis- und Dateinamen im Pfadnamen verwendet. Mit dem Semikolon (;) wird ein Kommentar eingeleitet, die folgenden Zeichen in dieser Zeile werden von AmigaDOS ignoriert. Der Stern (*) verweist auf das aktuelle Fenster. Es gab jedoch vereinzelt Programme, die den Stern als Jokerzeichen (ähnlich den PC's) verwendeten.

Mit dem umgekehrten Apostroph können innerhalb von Zeichenketten AmigaDOS-Befehle ausgeführt werden. Geben Sie beispielsweise

1> ECHO "Die Diskette in df0: enthält `dir df0:`"

ein, so wird erst der Text "Die Diskette in df0: enthält" ausgegeben und anschließend der Befehl *dir df0:* ausgeführt. Der umgekehrte Apostroph befindet sich auf der Taste links neben der "1".

3.4 - Die Jokerzeichen von AmigaDOS

Für besonders faule Programmierer (solche wie ich), gibt es die Möglichkeit, mit Hilfe von Jokerzeichen die Dateinamen in Argumenten abzukürzen, oder auch gleichzeitig mehrere Dateien anzusprechen. Folgende Zeichen werden von vielen AmigaDOS-Befehlen verwendet:

Steht für ein beliebiges einzelnes Zeichen. Zum Beispiel können mit dem Befehl "copy a?o to ram:" folgende Dateien kopiert werden: "a o", "aZo", "afo", usw.

#<m> In dem Dateinamen kann das Muster <m> null- oder mehrfach auftreten. Zum Beispiel steht "er#k" für "er", "erk", "erkk", "erkkk", usw. In Verbindung mit ? können Dateinamen abgekürzt werden. Eine Abkürzung bedeutet ja eigentlich nichts anderes, als daß nach den ersten Buchstaben eine beliebige Anzahl beliebiger Zeichen folgen kann. Folglich ist "Read#?" beispielsweise die Abkürzung von "ReadMe" oder "ReadDisk", usw.

% steht für eine Zeichenkette der Länge Null, also für kein Zeichen. Dieses auf den ersten Blick sicherlich sinnlos erscheinende Jokerzeichen gewinnt aber in Verbindung mit den beiden folgenden Zeichen an Bedeutung. Das Beispiel folgt deshalb auch erst weiter unten.

() Die Klammern "gruppieren Muster" (laut Benutzerhandbuch). Nun darunter ist nichts anderes zu verstehen, als daß mit Klammern mehrere Buchstaben zu einem Muster zusammengefaßt werden können. Beispielsweise steht "#(txt)" für eine beliebig ofte Wiederholung der Zeichenkette "txt" im Dateinamen, wie "DosKurs.txt" oder "Btxttxt" usw.

<m1>|<m2> Mit Hilfe dieses Zeichens werden zwei Muster mit "ODER" verknüpft. Folglich kann sich die Zeichenkette "run|copy|cd" sowohl auf die Datei "cd" als auch auf "run" oder "copy" beziehen.

Ich bin Ihnen noch ein Beispiel für das “%” schuldig:
Mit “Test(1%)#?.text” können folgende Dateien angesprochen werden: “Test1.text”, “Test.text”, “Testergebnisse.text”, usw.

Das Apostroph (Tastenkombination rechte Alt/ä) hebt die Bedeutung des Jokerzeichens auf. Dies ist dann sinnvoll, wenn der betreffende Dateiname selbst Jokerzeichen enthält.

Beispiel: Sollten von den beiden vorhandenen Dateien “Test#4” und “Test#44” nur die erstere kopiert werden, so ist bei <quelle> “Test’#4” einzugeben.

[<m1>-<m2>]Das Zeichen an dieser Stelle muß im Bereich <m1> bis <m2> liegen. Auch hier wird nicht zwischen Groß- und Kleinschreibung unterschieden. Wollen Sie beispielsweise die Dateien von “A” bis “D” ins RAM: kopieren, geben Sie folgende Zeile ein:

copy [a-d]#? to ram:

Im Grunde leisten die Jokerzeichen bei der Arbeit mit der Shell wertvolle Dienste, es gibt jedoch Situationen, in denen der Anwender diese Zeichen verwünschen möchte. Ein unüberlegter Gebrauch der Jokerzeichen in Verbindung mit Befehlen, die Daten löschen, kann verheerende Folgen haben. Vergewissern Sie sich unbedingt jedesmal, wenn Sie mit einem einzigen Delete-Befehl mehrere Dateien entfernen wollen, daß Sie wirklich nur die gewünschten Dateien ansprechen und keine weitere - und die Groß- und Kleinschreibung wird immer noch nicht berücksichtigt!

3.5 - Logische und physikalische Geräte

Da viele Programme immer wieder die gleichen Daten benötigen, ist es natürlich erstrebenswert, diese Daten in Verzeichnissen zu sammeln, die dann einen festgelegten Platz auf den jeweiligen Datenträgern haben. So müßte jedes Programm in der Lage sein, sich zum Beispiel die benötigten Zeichensätze aus einem eigens dafür geschaffenen Verzeichnis zu holen, wenn der Name dieses Verzeichnisses und der Pfad dorthin genormt ist.

Da aber auf der anderen Seite die Daten beispielsweise beim Kopieren einer Diskette auf die Festplatte notgedrungen in andere Verzeichnisse gelangen, ist die Festlegung von Verzeichnissen selbst sinnlos. Daher hat man eine Lösung gesucht, die sowohl die Namen vereinheitlicht als auch die unterschiedlichen Pfade der jeweiligen Daten berücksichtigt: die logischen Geräte.

Es gibt folgende von Commodore festgelegte Namen für logische Geräte:

Die logischen Geräte:

- ENV:** Bezeichnet ein Verzeichnis, in das Daten kommen, die für die Arbeitsumgebung (Shell/CLI, aber auch von anderen Programmen) benötigt werden. Seit der Betriebssystemversion 3.0 werden hier im Verzeichnis "sys" die aktuellen Voreinstellungen gespeichert, die bei jedem Neustart des Amigas automatisch berücksichtigt werden.
- T:** Steht für ein Verzeichnis, in dem Editoren Zwischenspeicher angelegen können. Auch der AmigaDOS-Befehl Execute legt Daten darin ab, um beispielsweise die Abarbeitung der Startup-Sequence zu beschleunigen.
- FONTS:** In diesem Verzeichnis befinden sich normalerweise die zur Verfügung stehenden Zeichensätze (engl: fonts). Ist auf dem Datenträger, von dem gebootet wurde, ein gleichnamiges Verzeichnis bereits vorhanden, so wird automatisch ein logisches Gerät mit diesem Verzeichnis geschaffen.
- S:** Hier stehen im Normalfall Script-Dateien (Auch Befehls- oder Batchdateien genannt), beim Booten wird automatisch dieses logische Gerät definiert, wenn ein Verzeichnis s vorhanden ist.
- L:** Für diverse Hilfsmittel (Handler, Disk-Validator usw.) ist dieses logische Gerät geschaffen worden, das ebenfalls beim Booten definiert wird.

- C:** Hier befinden sich alle AmigaDOS-Befehle. Auch dieses logische Gerät wird beim Booten vom Amiga selbstständig installiert, gleichzeitig ist dieses logische Gerät automatisch auch im Suchpfad für Befehle aufgeführt.
- DEVS:** Seltener benötigte Devices werden aus diesem Verzeichnis geladen.
- LIBS:** Wenn Libraries gebraucht werden, sind sie nur hier zu finden.
- SYS:** Bezeichnet den Datenträger, von dem aus gebootet wurde.
- PIPE:** Dieses Gerät dient der zeitweiligen Speicherung von Daten. Es können hier wie in normalen physikalischen Datenträgern Dateien angelegt werden, die alle einzeln über Namen angesprochen werden können. Es ist jedoch nicht möglich, Verzeichnisse anzulegen. Das besondere an diesem Gerät ist, das die Daten automatisch gelöscht werden, sobald sie einmal wieder aus der Pipe geholt wurden. Das Gerät eignet sich optimal zur direkten Übertragung von Daten zwischen Programmen.
- HELP:** Die Startup-Sequence sorgt dafür, daß dieses Logische Gerät immer auf das Verzeichnis "Help" eingestellt ist, das selbst als logisches Gerät "LOCALE:" definiert ist. Programme können sich hier gegebenenfalls Anleitungen in der jeweils voreingestellten Landessprache holen.
- LOCALE:** Hier sind alle Informationen zu den Landesspezifischen Voreinstellungen zu finden, die beispielsweise im Programm "Locale" der Prefs-Schublade benötigt werden.
- KEYMAPS:** Hier sind die Tastaturbelegungen untergebracht.
- PRINTERS:** Analog zu den Tastaturbelegungen sind im Verzeichnis dieses logischen Gerätes alle verfügbaren Druckertreiber verstaut.
- REXX:** Die originale Startup-Sequence sorgt dafür, daß über dieses logische Gerät das Verzeichnis REXXC der Workbench angesprochen werden kann.

- CLIPS:** Ebenfalls von der Startup-Sequence aus wird hiermit das Verzeichnis "RAM:Clipboards" angesprochen.
- ENVARC:** Dieses Logische Gerät gehört normalerweise zum Verzeichnis "Workbench:Prefs/Env-Archive". Es beinhaltet die Voreinstellungen, die bei jedem Bootvorgang automatisch berücksichtigt werden.

Neben diesen festgelegten logischen Geräten kann man natürlich auch noch weitere definieren. Sinn und Zweck dieser Entwicklung ist, den Zugriff auf benötigte Daten dadurch zu vereinfachen, daß das Programm nur den Namen des logischen Gerätes vor die gewünschte Datei stellen muß. Die Verbindung zwischen dem logischen Gerät und der tatsächlichen Stelle im beliebigen Verzeichnis wird dann vom System über eine Liste hergestellt, die mit dem Befehl ASSIGN (siehe auch dort) erstellt und bearbeitet werden kann. Man kann dadurch die logischen Geräte den jeweiligen Gegebenheiten anpassen, die Programme finden trotzdem die benötigten Daten.

Es gibt weiterhin aber auch folgende physikalische Geräte, die nicht festgelegt oder geändert werden können:

- NIL:** Dieses Gerät bezeichnet das "Nichts". Es hat die Eigenschaft, daß es nur beschreibbar ist, aber nicht davon gelesen werden kann. Es wird vor allem dann verwendet, wenn irgendwelche Meldungen unerwünscht sind. Diese werden mit der Ausgabeumleitung ins Nichts unterdrückt.

Beispiel: 1> dir >nil:

- DFx:** Hierzu muß wohl am wenigsten geschrieben werden, es sind die Bezeichnungen der einzelnen Diskettenlaufwerke DF0: bis DF3:

- HDx:** Bezeichnet analog zu den DFx: die einzelnen Festplatten (bzw. deren Partitionen).

- PCx:** Dieses Gerät ist eine Besonderheit seit der Version 2.1. Dieses Gerät muß erst installiert werden, bezeichnet aber dennoch ein physikalisches Gerät. Hiermit wird das Diskettenlaufwerk als 720KByte-Laufwerk für MS-DOS-Disketten angesprochen.

- PRT:** Dieses Gerät bezeichnet den Drucker, es kann nur beschrieben werden. Erwähnenswert ist hier, daß die Daten erst noch durch den Druckertreiber müssen, bevor sie an den Drucker gelangen, wenn sie zum Gerät PRT: geschickt werden. Die Druckertreiber wandeln in der Regel die Escape-Sequenzen die oben aufgeführt sind in für den Drucker verständliche Befehle um.
- PAR:** Normalerweise bezeichnet auch dieses Gerät den Drucker (der in der Regel am Parallelport angeschlossen ist). Hier werden die Daten jedoch unbearbeitet an den Drucker weitergegeben, ohne den Umweg über den Druckertreiber zu gehen. Da aber diese Schnittstelle auch als Eingang benutzt werden kann, können hier - sofern entsprechende Geräte angeschlossen sind - auch Daten gelesen werden.
- SER:** Dieses Gerät bezeichnet die serielle Schnittstelle. Hier können Modems oder MIDI-Interfaces angeschlossen und damit über SER: angesprochen werden.
- AUX:** Auch dieses Gerät bezieht sich auf die serielle Schnittstelle, hier werden die Daten allerdings nicht gepuffert, d.h. es existiert kein Zwischenspeicher, der die Datenübertragung beschleunigt.
- CON:** Dieses Gerät bezeichnet ein Konsolenfenster, wie wir es von der Shell her schon kennen.
- RAW:** Dieses Gerät bezeichnet ein Konsolenfenster - ähnlich dem Shell-Fenster. Es bestehen jedoch Unterschiede in der Art, wie Eingaben und Ausgaben vorallem von nichtdruckbaren Zeichen verarbeitet werden. Hier besteht keine Möglichkeit, die Zeilen zu bearbeiten, alle Zeichen wie beispielsweise die Cursor-Tasten oder die Ctrl-Taste werden direkt an den für das Fenster verantwortlichen Prozeß übergeben.
- CC0:** Dieses Gerät bezeichnet die Schnittstelle für PCMCIA-Karten auf der linken Seite des Amiga 1200 und des Amiga 600.

- RAM:** Das Gerät RAM: bezeichnet einen diskettenähnliche Datenträger im Arbeitsspeicher, der immer so groß ist, wie die darin enthaltenen Daten. Im RAM: können auch Verzeichnisse verwaltet werden. Der Inhalt von RAM: geht bei jedem Neustart des Amigas verloren.
- RAD:** Dies ist eine RAM-Disk, die einen festgelegten Speicherbereich in der gleichen Größe wie eine 880-KByte-Diskette belegt. Dieser Speicherbereich ist für das Betriebssystem nicht verfügbar, solange die RAD: aktiv ist, auch dann nicht, wenn sie nicht komplett gefüllt ist. Das Besondere an RAD: ist, daß der Inhalt bei einem Reboot (etwa über die drei Tasten Amiga-Amiga-Ctrl) des Amigas nicht verloren geht, sondern nur beim Ausschalten des Rechners. Der Amiga kann sogar von dieser Diskette booten, falls alle notwendigen Daten in der RAD: vorhanden sind.

PAG Sandini

3.6 - Die AmigaDOS-Befehle

Jetzt geht es zur Beschreibung der AmigaDOS-Befehle im Einzelnen. Dabei werden die einzelnen Befehle alphabetisch geordnet mit allen zugehörigen Parametern und Argumenten beschrieben.

3.6.1 - Die Befehlskonventionen und Schablone

Jeder Befehl wird nach folgenden Schema beschrieben:

- Syntax:** Hier wird die genaue Syntax mit allen möglichen Argumenten und Parametern aufgeführt. Sollten in den einzelnen Workbenchversionen Unterschiede vorhanden sein, wird die Syntax für jede Version getrennt aufgeführt.
- Schablone:** Die Schablone stellt in einer Kurzform alle Möglichkeiten der Befehlssyntax dar. Auch werden Unterschiede zwischen den einzelnen Workbenchversionen berücksichtigt.
- Pfad:** Gibt an, ob der Befehl fest in die Shell integriert ist, bzw. in welchen Verzeichnis das entsprechende Programm zu finden ist.
- Funktion:** kurze Beschreibung der Funktionsweise des Befehls
- Beschreibung:** ausführliche Beschreibung des Befehls
- Argumente:** Beschreibung der Funktionen der einzelnen Parameter.
- Beispiele:** Da oft Beispiele mehr sagen können als tausend Worte, sind sie bei der Befehlsbeschreibung unumgänglich.
- Bemerkungen:** Bei einigen Befehlen sollten interessante Zusatzinformationen dem Leser natürlich nicht vorenthalten bleiben. Sie werden im Normalfall unter den Bemerkungen erwähnt.

Siehe auch: Da ein Befehl in den seltensten Fällen allein steht, kann seine Wirkung erst im Gesamtüberblick richtig eingeschätzt werden. Daher wird hier meist auf andere verwandte Befehle hingewiesen.

Zunächst sollten noch einige Vereinbarungen zur Darstellung der Befehlsyntax getroffen werden, die im gesamten Kapitel beibehalten werden (und auch mit den Benutzerhandbüchern von Commodore übereinstimmt).

GROSS geschriebene Wörter sind Schlüsselwörter, sie müssen wieder genau so eingegeben werden, da sie in der Regel spezielle Bedeutungen tragen.

<> In spitzen Klammern stehen Argumente, an deren Stelle vom Anwender bestimmte Angaben übergeben werden müssen. Häufig wird beispielsweise die Darstellung <Datei> dafür stehen, daß an dieser Stelle ein Dateiname übergeben werden muß. Steht ein Argument in spitzen Klammern <> nicht innerhalb eines weiteren eckigen Klammernpaares [], so ist dieses Argument notwendig für diesen Befehl, andernfalls kann dieser nicht korrekt abgearbeitet werden.

[] In eckigen Klammern stehen Argumente, die angegeben werden können, aber nicht unbedingt angegeben werden müssen. Sie beeinflussen in der Regel die Arbeitsweise des Befehls.

{ } Stehen Argumente in geschweiften Klammern, können an dieser Stelle beliebig viele Argumente eingegeben werden.

| Sollten eine Auswahl mehrerer verschiedener Optionen (meist hinter dem Schlüsselwort "OPT") vorhanden sein, so werden diese Optionen durch einen senkrechten Strich "|" voneinander getrennt. Sie können in diesem Fall eine beliebige Kombination dieser Optionen dem Befehl übergeben.

Da die Befehlsschablone in den einzelnen Befehlen integriert ist, sollte auch diese etwas gewöhnungsbedürftige Darstellung näher erläutert werden. In einer Befehlsschablone werden die einzelnen Argumente durch Kommata voneinander getrennt. Jedem Argument folgt ein oder mehrere Schrägstriche und ein sogenannter Kennbuchstabe, der die Art des Argumentes wiedergibt.

Dieser Kennbuchstabe dient einzig und allein der Angabe der Art des jeweiligen Argumentes und ist nicht Teil des Befehls, darf folglich auch nicht eingegeben werden. Folgende Aufzählung zeigt die Bedeutung der einzelnen Kennbuchstaben:

- Argument/A** Dieses Argument muß immer angegeben werden.
- Option/K** Wollen Sie hier ein Argument übergeben, muß erst das angegebene Schlüsselwort (hier symbolisch "OPTION") eingegeben werden, gefolgt von dem eigentlichen Argument. Schlüsselwort und Argument sind durch mindestens ein Leerzeichen voneinander getrennt.
- Option/S** Es handelt sich bei OPTION um ein Schlüsselwort, das als Schalter fungiert und daher keine weiteren Argumente benötigt.
- Wert/N** An dieser Stelle muß eine Zahl eingegeben werden.
- Argument/M** An dieser Stelle werden auch mehrere Argumente dieser Art akzeptiert. Dies entspricht der Darstellung der Argumente in geschweiften Klammern bei der Syntax-Beschreibung. Die Anzahl der Argumente ist lediglich durch die maximale Zeilenlänge beschränkt, die Mehrfachargumente müssen jedoch vor allen anderen Parametern eingegeben werden.
- Zeichenkette/F** Diese Zeichenkette schließt die Befehlseingabe ab. Was dann nach dieser Zeichenkette folgt, wird vom Befehl als ein Wort interpretiert, auch wenn darin mehrere Leerzeichen enthalten sind. Es werden keine Anführungsstriche benötigt, wie dies sonst üblich ist.
- =** Es gibt Schlüsselwörter, die sowohl in Kurzform als auch ausgeschrieben von einem Befehl akzeptiert werden. Sind zwei Schreibweisen für ein und denselben Schalter gültig, werden beide von einem Gleichheitszeichen getrennt in der Schablone dargestellt, gefolgt vom Kennbuchstaben für die Argumentenart. Das Gleichheitszeichen es ebenso wie der Schrägstrich nicht Teil des Befehls und darf ebenfalls nicht eingegeben werden.

Es ist jedoch möglich und oft auch sehr sinnvoll, den einzelnen Schlüsselwörtern mit Gleichheitszeichen die Argumente zuzuordnen. So sind in umfangreicheren Befehlszeilen eindeutige Zuweisungen sichergestellt.

Beispiel:

CHANGETASKPRI pri=5 prozess=3

Mit diesen Vorbereitungen sind wir jetzt gerüstet, uns in die Menge der AmigaDOS-Befehle zu stürzen.

PAG Sandini

3.6.2 - Ein erster Überblick

Die folgende Liste zeigt, in welcher Version der Workbench seit Kickstart 2.04 welche Befehle verfügbar sind. Dabei bedeuten:

- Befehl ist nicht verfügbar.
- + Befehl ist verfügbar.
- v Befehl ist gegenüber früheren Versionen verbessert worden.

BEFEHL	2.0	2.1	3.0
ADDBUFFERS	v	+	+
ADDDATATYPE	-	-	+
ADDMONITOR	+	-	-
ALIAS	v	+	v
ASK	+	+	+
ASSIGN	v	+	+
AUTOPOINT	+	+	+
AVAIL	v	+	+
A2024	-	+	+
BINDDRIVERS	+	+	+
BLANKER	+	v	+
BREAK	+	+	+
BRU	+	+	+
CALCULATOR	+	v	+
CD	v	+	+
CHANGETASKPRI	+	v	+
CLICKTOFRONT	+	+	+
CLOCK	+	v	+
CMD	+	v	+
COLORS	+	+	+
CONCLIP	+	+	+
COPY	v	+	+
CPU	+	+	+
CROSSDOS	-	+	+
DATE	+	+	+
DBLNTSC	-	-	+
DBLPAL	-	-	+
DELETE	v	+	+
DIR	v	+	+
DISKCHANGE	+	+	+

AMIGA DOS 3.0

DISKCOPY	+	+	+
DISKDOCTOR	+	-	-
DISPLAY	+	-	-
DPAT	v	+	+
ECHO	v	+	+
ED	v	+	+
EDIT	v	+	+
ELSE	v	+	+
ENDCLI	+	+	+
ENDSHELL	v	+	+
ENDIF	v	+	+
ENDSKIP	+	+	+
EVAL	v	+	+
EXCHANGE	+	v	+
EXECUTE	v	+	+
FAILAT	v	+	v
FAULT	+	+	+
FILENOTE	v	+	+
FIXFONTS	+	+	+
FKEY	+	v	+
FONT	+	+	+
FORMAT	v	v	v
GET	+	+	+
GETENV	v	+	+
GRAPHICDUMP	+	+	+
ICONTROL	+	v	+
ICONX	+	+	+
IF	v	+	+
IHELP	+	-	-
INFO	+	+	+
INITPRINTER	+	+	+
INPUT	+	+	+
INSTALL	v	+	+
IPREFS	+	+	+
JOIN	v	+	+
LAB	v	+	+
LIST	v	+	+
LOADWB	+	v	+
LOCALE	-	+	+
LOCK	v	+	+
MAGTAPE	+	+	+
MAKEDIR	v	+	+

MAKELINK	+	+	+
MEMACS	+	+	+
MORE	+	+	+
MOUNT	v	+	+
MOUSEBLANKER	-	+	+
MULTISCAN	-	+	+
MULTIVIEW	-	-	+
NEWCLI	v	+	+
NEWSHELL	v	+	+
NOCAPSLOCK	+	+	+
OVERSCAN	+	v	+
PAL	-	+	+
PALETTE	+	+	v
PARK	+	+	+
PATH	v	v	+
PCD	v	+	+
POINTER	+	v	+
PREPCARD	-	+	+
PRINTER	+	v	+
PRINTERGFX	+	v	+
PRINTERPS	+	v	+
PRINTFILES	+	+	+
PROMPT	v	+	+
PROTECT	v	+	+
QUIT	v	+	+
RELABEL	+	+	+
REMRAD	v	+	+
RENAME	+	+	+
REQUESTCHOICE	-	-	+
REQUESTFILE	-	-	+
RESIDENT	v	+	+
REXXMAST	+	+	+
RUN	+	+	v
SAY	+	+	+
SCREENMODE	+	+	+
SEARCH	v	+	+
SERIAL	+	v	+
SET	+	+	+
SETCLOCK	+	+	+
SETDATE	v	v	+
SETENV	v	+	+
SETFONT	+	+	v
SETKEYBOARD	-	+	+

AMIGA DOS 3.0

SETMAP	+	-	-
SETPATCH	v	v	+
SHOWCONFIG	-	+	+
SKIP	+	+	+
SORT	v	+	+
SOUND	-	+	+
SPAT	v	+	+
STACK	+	+	+
STATUS	+	+	+
SUPER72	-	+	+
TIME	+	v	+
TYPE	v	+	+
UNALIAS	+	+	+
UNSET	+	+	+
UNSETENV	+	+	+
VERSION	v	v	+
WAIT	+	+	v
WBPATTERN	+	v	+
WHICH	v	+	+
WHY	+	+	+
:	+	+	+
<	+	+	+
>	+	+	+
>>	+	+	+

ADDBUFFERS

Syntax:

2.0/2.1/3.0: ADDBUFFERS [DRIVE]<drive>: [[BUFFERS]<n>]

Schablone:

2.0/2.1/3.0: DRIVE/A, BUFFERS/N

Pfad:

C:

Funktion:

Die Anzahl der Pufferspeicher für ein Diskettenlaufwerk wird erhöht.

Beschreibung:

Mit dem Befehl **ADDBUFFERS** wird die Anzahl der Pufferspeicher für ein bestimmtes Diskettenlaufwerk um n geändert. Positive n erhöhen die Zahl der Speicher, negative Zahlen verringern die Anzahl. Ein größerer Puffer kann die Zugriffszeit auf Daten der Disketten erheblich verkürzen. Jeder weitere Speicher verkleinert den verfügbaren Arbeitsspeicher um 512 Byte.

Voreingestellt sind für ein Diskettenlaufwerk 20 Speicher, für eine Festplatte 30 Speicher. Diese Werte liegen erfahrungsgemäß im günstigsten Bereich. Werden weniger Speicher zur Verfügung gestellt, verlängert sich die Zugriffszeit erheblich, bei einer größeren Anzahl wird dem System jedoch unnötig viel Speicher entzogen, wodurch Programme sehr stark beeinträchtigt werden können. So sollte bei der Bemessung der Speicheranzahl immer die Größe des verfügbaren RAM einkalkuliert werden.

Wird bei **ADDBUFFERS** lediglich die Laufwerksbezeichnung <drive>: eingegeben, ohne den Speicher zu verändern, wird die aktuelle Anzahl der diesem Laufwerk zugewiesenen Speicher angezeigt.

Argumente:

DRIVE/N hier ist anzugeben, für welches Laufwerk die Anzahl der Pufferspeicher angezeigt oder geändert werden soll.

BUFFERS/A Diese Zahl gibt an, wie die Gesamtzahl der Pufferspeicher geändert werden soll. Fehlt dieses Argument, wird die aktuelle Zahl der dem Laufwerk zugewiesenen Pufferspeicher angezeigt.

Beispiele:

1.Workbench:> ADDBUFFERS df0:
df0: has 20 buffers

Ohne Zahlenangabe wird die aktuelle Anzahl der dem Laufwerk df0: zugewiesenen Speicher angezeigt: df0: hat 20 Speicher.

1.Workbench:> ADDBUFFERS df0: 13
df0: has 33 buffers

Die Anzahl der Pufferspeicher für das Laufwerk df0: wird um 13 auf 33 erhöht...

PAG Sandini

1.Workbench:> ADDBUFFERS df0: -12
df0: has 21 buffers

... anschließend wieder um 12 auf 21 verringert.

Bemerkungen:

Auch wenn das neue FFS (FastFileSystem) zu deutlichen Steigerungen der Zugriffszeiten auch auf Disketten ermöglicht, sollten die Möglichkeiten der Diskettenbeschleunigung durch die Zuweisung von mehr Pufferspeicher für die Laufwerke nicht außer Acht gelassen werden. Denn unabhängig davon, wie schnell physikalische Massenspeicher werden können, ist die Zugriffszeit auf Daten im RAM naturgemäß immer kürzer.

Siehe auch:

COPY

ADDDATATYPES

Syntax:

2.0/2.1: Befehl nicht implementiert

3.0: ADDDATATYPES [[FILE] { file|pattern }] [QUIET] [REFRESH]

Schablone:

2.0/2.1: Befehl nicht implementiert

3.0: FILE/M,QUIET/S,REFRESH/S

Pfad:

C:

Funktion:

Die Liste der Datentypen wird für die datatype.library erweitert.

Beschreibung:

PAG Sandini

Dieser Befehl wird vor allem von der System-Software oder von Installationsprogrammen genutzt und dient der Erweiterung der Liste der Dateientypen für die "datatypes.library". Die Beschreibungen werden im Verzeichnis "DEVS:DataTypes" gespeichert.

Argumente:

FILE/M hier können eine oder mehrere Dateien angegeben werden, die dann die Beschreibungen für weitere Dateientypen enthalten, um die die Liste erweitert werden soll.

QUIET/S Dieser Schalter unterdrückt die Ausgabe von Meldungen.

REFRESH/S Mit dieser Option wird das Verzeichnis "DEVS:DataTypes" auf Änderungen hin untersucht.

ADDMONITOR

Syntax:

2.0: ADDMONITOR [NUMBER=]<num> [NAME=]<name>
 [HBSTRT=<n>] [HBSTOP=<n>] [HSSTRT=<n>]
 [HSSTOP=<n>] [VBSTRT=<n>] [VBSTOP=<n>]
 [VSSTRT=<n>] [VSSTOP=<n>] [MINROW=<n>]
 [MINCOL=<n>] [TOTROWS=<n>] [TOTCLKS=<n>]
 [BEAMCON0=<n>]

2.1/3.0: Befehl durch andere im Verzeichnis DEVS:Monitors ersetzt.

Schablone:

2.0: NUM/A/N,NAME/A,HBSTRT/K,HBSTOP/K,HSSTRT/K,
 HSSTOP/K,VBSTRT/K,VBSTOP/K,VSSTRT/K,VSSTOP/K,MIN
 ROW/K, MINCOL/K,TOTROWS/K,TOTCLKS/K,BEAMCON0/K

2.1/3.0: Befehl durch andere im Verzeichnis DEVS:Monitors ersetzt.

Pfad:

SYS:System

Funktion:

Mit ADDMONITOR werden der Monitorliste im Verzeichnis DEVS:Monitors weitere Einträge hinzugefügt.

Beschreibung:

Mit diesem Befehl wird die Liste der verfügbaren Monitor-Einstellungen im Verzeichnis DEVS:Monitors um einen weiteren Eintrag ergänzt. Dazu werden alleine die beiden ersten Argumente benötigt. Die Wirkung ist identisch mit der Verschiebung neuer Monitor-Symbole in die entsprechende Schublade der Workbench. Anschließend muß mit dem Programm ScreenMode der Preferences-Schublade der neue Monitor ausgewählt werden, um seine Darstellungsmodi nutzen zu können. Man kann ADDMONITOR auch benutzen, um bestimmte Hardwareregistereinträge, die sich auf den Bildschirmmodus beziehen, zu ändern. Diese Möglichkeit erfordert eine sehr

gute Kenntnis des Systems und sollte daher absoluten Fachleuten vorbehalten bleiben. Die hierfür angegebenen Argumente werden direkt in Hardwareregister übertragen und sollten daher nur mit größter Vorsicht verwendet werden.

Argumente:

- NAME/A** Name steht für die Bezeichnung des Monitors (NTSC, PAL, Multiscan oder A2024). Beispiel: ADDMONITOR NUM=2 NAME=Pal
- NUM/A/M** Diese Nummer von 1 bis 4 wird dem Monitortyp zugewiesen. Beispiel: ADDMONITOR NUM=3 NAME Multiscan
- BEAMCON0/K** Eintrag in das Beam Control Register 0. (Dieses Schlüsselwort endet mit einer Null und nicht mit dem Buchstaben O). Der hier angegebene Wert einer 16-Bit-Zahl wird direkt in das Kontroll-Register geschrieben, ohne die Werte auf ihre Zulässigkeit zu überprüfen. Ein falscher Wert kann theoretisch den angeschlossenen Monitor zerstören! Experimentieren Sie also nicht mit diesem Argument. Folgende Tabelle gibt einen groben Überblick über die Bedeutung der einzelnen Bits (Angaben ohne Gewähr):

Bit-Nr.	Funktion
15	unbenutzt
14	enable software blanking position
13	ignore latched pen
12	use values from VBSTRT/VBSTOP
11	disable long/short line toggle
10	redirect composite H and V sync
9	enable values from VSSTRT/VSSTOP
8	enable values from HSSTRT/HSSTOP
7	enable variable beam counter
6	enable special mode
5	enable programmable PAL mode
4	enable values from VSync and HSync registers
3	enable redirection of composite blank
2	control neg/pos going Comp sync
1	control neg/pos going VSync
0	control neg/pos going HSync

HBSTRT/K	Spalte, in der die horizontale Anzeige beginnt
HBSTOP/K	Spalte, in der die horizontale Anzeige endet
HSSTRT/K	Spalte, in der die horiz. Sync startet
HSSTOP/K	Spalte, in der die horiz. Sync endet
VBSTRT/K	Zeile, in der die vertikale Anzeige beginnt
VBSTOP/K	Zeile, in der die vertikale Anzeige endet
VSSTRT/K	Zeile, in der die vert. Sync startet
VSSTOP/K	Zeile, in der die vert. Sync endet
MINCOL/K	Minimale Anzahl der Bildschirmspalten
MINROW/K	Minimale Anzahl der Bildschirmzeilen
TOTCLKS/K	Gesamtzahl der Clock-Zyklen
TOTROWS/K	Gesamtzahl der Bildschirmzeilen.

Bemerkung:

Verwenden Sie diesen Befehl nur zur Erweiterung der Monitor-Liste. Ein unsachgemäßer Gebrauch kann zur Zerstörung des angeschlossenen Monitors oder Fernsehers führen.

ALIAS

Syntax:

2.0/2.1/3.0: ALIAS [[NAME]<name>] [[STRING]<string>]

Schablone:

2.0/2.1: NAME,STRING

3.0: NAME,STRING/F

Pfad:

in die Shell integriert

Funktion:

Legt Synonyme für Befehle fest oder zeigt gültige Festlegungen an.

Beschreibung:

Wie der Name schon erkennen läßt, ermöglicht dieser Befehl die Umbenennung von AmigaDOS-Befehlen oder häufig benutzten Ausdrücken. Dieser im Grunde sehr leicht verständliche Befehl kann dem Anwender sehr viel lästiger Tipparbeit sparen, in dem für oft benutzte Befehle Abkürzungen vereinbart werden. Die Festlegung gilt in der Regel nur für den Shell-Prozess, in dem diese Vereinbarung getroffen wurde, in anderen Shell-Fenstern wird sie nicht verwendet. Es ist daher empfehlenswert diese Festlegungen in der Datei s:Shell-Startup vorzunehmen, da die darin enthaltenen Befehle bei jedem Neustart einer Shell automatisch ausgeführt werden.

Wird in einer Befehlszeile der <Name> eingegeben, so wird durch den ALIAS-Befehl dieser an jeder Stelle durch die Zeichenkette <String> ersetzt, bevor die Befehlszeile vom AmigaDOS selbst ausgewertet wird. Es kann kein einzelnes Argument eines Befehles durch eine ALIAS-Umbenennung ersetzt werden.

Es ist jedoch möglich innerhalb des neuen Synonyms nachträglich weitere Argumente einzufügen. Die Stelle, an der diese Argumente eingefügt werden sollen, muß mit einem eckigen Klammerpaar [] gekennzeichnet werden.

Argumente:

NAME das neue Synonym, das festgelegt werden soll. Beachten Sie, daß auch dieser Name keine Leerzeichen enthalten sollte.

STRING die Zeichenkette, durch die das neue Synonym vor jeder Befehlsausführung ersetzt werden soll. Bei den Versionen 2.0 und 2.1 muß diese Zeichenkette in Anführungsstriche eingeschlossen werden, falls Leerzeichen enthalten sind. Seit der Version 3.0 sind die Anführungszeichen nicht mehr notwendig.

Beispiele: Sie können nach dem Befehl

```
1> ALIAS XCOPY "copy CLONE"
```

den Befehl "Copy Clone datei to verzeichnis" immer mit "xcopy datei to verzeichnis" abkürzen.

In der Zeile

```
1> LIST df0: to RAM:Inhalt LFORMAT "copy %S%S to hd1:Tools"
```

kann aber das LFORMAT-Argument nicht mit ALIAS durch ein Synonym ersetzt werden.

Wenn Sie mit dem Befehl

```
1> ALIAS da "DIR [] ALL"
```

das neue Synonym "da" definiert haben, können Sie den kompletten Inhalt des Verzeichnisses "RAM:Test" durch folgenden Befehl anzeigen lassen:

```
1> da ram:test
```

Diese Zeile wird vor der Ausführung für das AmigaDOS in den Befehl "DIR ram:test ALL" umgewandelt.

Bemerkungen:

Eine mit dem ALIAS-Befehl vorgenommene Umbenennung wird bei der Auswertung einer Befehlszeile zuerst ersetzt. Diese Eigenschaft kann man beispielsweise auch nutzen, um bestimmte Befehle vor unbefugter Benutzung zu sperren. Geben Sie beispielsweise die Zeile

```
1> ALIAS FORMAT " "
```

ein, so bewirkt jede Eingabe des Befehls **FORMAT** eine Umwandlung in ein Leerzeichen, so daß dieser Befehl nicht ausgeführt werden kann. Eine kleine Anregung:

1> **ALIAS FORMAT ECHO** "Das darfst Du nicht"

Natürlich kann diese Sperre durch einen weitere Alias-Definition oder mit **UNALIAS** aufgehoben werden.

Siehe auch:

ASSIGN, UNALIAS

PAG Sandini

ASK

Syntax:

2.0/2.1/3.0: ASK [PROMPT] <"prompt">

Schablone:

2.0/2.1/3.0: PROMPT/A

Pfad:

in die Shell integriert

Funktion:

Während der Abarbeitung einer Befehlsdatei kann eine Entscheidungsfrage gestellt werden.

Beschreibung:

Dieser Befehl ist nur in einer Befehlsdatei sinnvoll. Dabei wird die Eingabeaufforderung <PROMPT> angezeigt, und gewartet, bis der Benutzer die Frage entweder mit "Y" (YES=ja) oder "N" (NO = Nein) beantwortet. Wird nur die RETURN-Taste gedrückt, wird dies wie die Eingabe von "N" gewertet.

Die Frage sollte also stets so formuliert werden, daß Sie klar mit Ja oder Nein beantwortet werden kann, andere Antworten führen zu einer Wiederholung der Frage. Abhängig davon, welche Antwort eingegeben wurde, wird die Bedingungsvariable der Shell auf den Wert 5 (WARN) bei "Y" oder 0 bei "N" gesetzt.

Mit einer anschließenden IF-Anweisung kann diese Bedingungsvariable abgefragt und ausgewertet werden.

Argument:

PROMPT/A Hier muß die Eingabeaufforderung eingegeben werden soll.
Wenn die Frage Leerzeichen enthält, muß die Eingabeaufforderung in Anführungszeichen gesetzt werden.

Beispiel:

In einer Befehlsdatei stehen folgende Zeilen:

```
ASK "Dateien kopieren? (Y/N)"
```

```
IF WARN
```

```
COPY df0:#? TO ram:
```

```
ELSE
```

```
ECHO "Dateien wurden nicht kopiert!"
```

```
ENDIF
```

Es wird die Frage gestellt: Dateien kopieren? (Y/N) Wird mit Y geantwortet, werden die Daten der Diskette in df0: ins Ram: kopiert, andernfalls erscheint die Meldung "Dateien wurden nicht kopiert!".

Siehe auch:

IF, ELSE und ENDIF

PAG Sandini

ASSIGN

Syntax:

2.0/2.1/3.0: ASSIGN [[NAME] <name>:] [[TARGET] {dir}] [LIST]
[EXISTS] [REMOVE] [DISMOUNT] [DEFER] [PATH]
[ADD] [VOLS] [DIRS] [DEVICES]

Schablone:

2.0/2.1/3.0: NAME, TARGET/M, LIST/S, EXISTS/S, REMOVE/S, DISMOUNT/S,
DEFER/S, PATH/S, ADD/S, VOLS/S, DIR/S, DEVICES/S

Pfad:

C:

Funktion:

Zuweisung von logischen Geräten an Verzeichnisse physikalischer Datenträger.

Beschreibung:

ASSIGN ist einer der wichtigsten AmigaDOS-Befehle und einer der mächtigsten. In erster Linie schafft er ein logisches Gerät, über das ein bestimmtes Verzeichnis wesentlich komfortabler weil wesentlich kürzer angesprochen werden kann. AmigaDOS kann jedes Verzeichnis als logisches Gerät interpretieren und entsprechend behandeln.

Es gibt verschiedenen Möglichkeiten in der Befehlssyntax und daraus resultierend auch mannigfaltige Einsatzmöglichkeiten. Ohne irgendwelche weiteren Angaben listet ASSIGN nur die aktuellen Zuweisungen auf, die entweder vom System automatisch beim Booten oder durch zwischenzeitliche Zuweisungen (etwa in der Startup-Sequence u.ä.) definiert wurden. Dabei wird die Anzeige unterteilt in die sogenannten "Volumes" (verfügbare oder zumindest namentlich bekannte Disketten und Festplatten/-partitionen), "Directories" (alle Verzeichnisse, denen logische Geräte zugeordnet wurden) und "Devices" (physikalische Geräte mit und ohne Treiber).

Jede Diskette, die derzeit verfügbar ist, enthält als Anhang das Wort "[Mountet]". Es kann auch sein, daß AmigaDOS mehr Disketten bekannt

sind, als derzeit verfügbar, etwa weil von einer Diskette ein Programm gestartet wurde, daß immer noch läuft, und deren Diskette bereits wieder aus dem Laufwerk entfernt wurde oder weil ein logisches Gerät einem Verzeichnis auf einer bestimmte Diskette zugewiesen wurde, die anschließend aus dem Laufwerk genommen wurde. Der interessanteste Teil ist jedoch der mit den "Directories". Hier zeigt ASSIGN, welche Verzeichnisse als logische Geräte angesprochen werden können.

Der ASSIGN-Befehl bietet auch die Möglichkeit, verschiedene Verzeichnisse unter dem gleichen logischen Gerät anzusprechen. Wird ein logisches Gerät angesprochen, dem mehrere Verzeichnisse zugeordnet wurden, so werden die einzelnen Verzeichnisse in der Reihenfolge der Ausgabe von oben nach unten bezüglich der angegebenen Datei untersucht. In der Liste werden solche mehrfachen Zuweisungen durch das +-Zeichen am Zeilenbeginn gekennzeichnet.

Argumente:

NAME Das neue logische Gerät erhält diesen Namen (gefolgt von einem Doppelpunkt; siehe auch Kapitel "Logische und physikalische Geräte").

TARGET/M An dieser Stelle wird ASSIGN die Liste der zuzuweisenden Verzeichnisse übergeben. Diese Verzeichnisse sollten immer über den vollen Pfadnamen eingegeben werden, ASSIGN selbst speichert immer den vollen Pfadnamen, damit die zugewiesenen Verzeichnisse immer und von jedem aktuellen Verzeichnis aus eindeutig und mühelos aufgerufen werden können.

LIST/S Dieser Schalter ist normalerweise voreingestellt. Wenn Sie jedoch nur zu einer bestimmten Zuweisung die Information angezeigt haben wollen, können Sie diese mit

ASSIGN <logGer:> LIST EXISTS

anzeigen lassen. Wird das Schlüsselwort "EXISTS" nicht angegeben, wird die Zuweisung zu dem gewünschten logischen Gerät augenblicklich und unwiederruflich gelöscht.

- REMOVE/S** Wird hinter dem Namen des logischen Gerätes das Schlüsselwort REMOVE angegeben, so wird diese Zuweisung gelöscht. Dies ist auch dann der Fall, wenn ASSIGN nur mit den Namen des logischen Geräte als einziges Argument aufgerufen wird.
- EXISTS/S** Dieser Schalter bietet eine einfache Möglichkeit, zu prüfen, ob ein logisches Gerät schon existiert oder nicht. Wird in der Liste der Zuweisungen das entsprechende logische Gerät gefunden, so wird die shelleigene Bedingungsvariable auf den Wert 0 gesetzt, andernfalls erhält sie den Wert 5 für WARN. Die Bedingungsvariable kann dann mit der IF-Anweisung weiterverarbeitet werden.
- ADD/S** An eine bereits bestehende Zuweisung können weitere Verzeichnisse durch Angabe des Schlüsselwortes ADD zugefügt werden. Fehlt dieses Schlüsselwort und werden ASSIGN nur der Name des logischen Gerätes und die zuzuweisende Verzeichnisse als Argumente übergeben, so werden die bestehenden Zuweisungen durch die neuen ersetzt.
- DEFER/S** Dieser Schalter ist mit der mächtigste des ASSIGN-Befehls. Wird bei einer Zuweisung das Schlüsselwort DEFER angegeben, erfolgt die Zuweisung erst dann, wenn das logische Gerät zum ersten Mal angesprochen wird. Üblicherweise werden die Zuweisungen schon beim Aufruf des ASSIGN-Befehls auf ihre Gültigkeit überprüft und die angesprochenen Verzeichnisse dadurch endgültig festgelegt. So ist es möglich, Zuweisungen für Verzeichnisse auf Disketten zu erstellen, ohne die entsprechende Diskette schon beim ASSIGN-Befehl einlegen zu müssen. Die Diskette wird erst dann angefordert, wenn das logische Gerät zum ersten Mal angesprochen wird. Wird diese Zuweisung durch einen Aufruf festgelegt, behält sie jedoch den entsprechenden Pfad gespeichert, bis eine neue Zuweisung eine Änderung herbeiführt. In der Zuweisungsliste wird ein mit DEFER angegebener Pfad in spitzen Klammer <> eingeschlossen.

DISMOUNT/S Mit dem Zusatz DISMOUNT können Sie einen Datenträger oder ein Gerät aus der Liste der angemeldeten Geräte löschen. Es wird jedoch nicht der von dem entsprechenden Gerät belegte Speicherplatz freigegeben. Es gibt auch keine Möglichkeit mehr das Gerät wieder anzumelden, so daß ein Neustart nötig wird, falls dies gewünscht wird. Diese Option sollte nur von Softwareentwicklern benutzt werden, da ein unsachgemäßer Gebrauch zu Fehlern und auch zu Rechnerabstürzen führen kann.

PATH/S Dieses Schlüsselwort ist nicht minder mächtig wie DEFER, es besitzt auch eine ähnliche Funktion. Auch hier besteht solange keine direkte Zuweisung, solange der Name des logischen Gerätes noch nicht verwendet wurde. Anders als bei DEFER besteht eine feste Zuweisung nur solange, wie sie gebraucht wird. Wird sie nicht mehr verwendet, so verlischt die aktuelle Zuweisung und wartet wieder, bis erneut davon gebrauch gemacht wird. Hier wird also jedes auf den angegebenen Pfad passende Verzeichnis akzeptiert, solange AmigaDOS ein passendes Verzeichnis findet. In der Zuweisungsliste wird ein mit PATH angegebener Pfad in eckigen Klammern [] eingeschlossen.

DEVICES/S Hier wird nur die mit "Devices:" überschriebene Teilliste ausgegeben.

DIR/S Analog zu DEVICES wird nur der Teil mit den zugewiesenen Verzeichnissen ausgegeben.

VOL/S Auch hier gibts nichts neues, nur fehlen bei dieser Ausgabe sowohl die "Devices" und "Directories", was bleibt dann noch übrig? - Erraten die Datenträgernamen (Volumes).

Beispiele:

Die alleinige Eingabe des Befehls ASSIGN zeigt die komplette Liste an:

1.Workbench:> assign

Volumes:

Ram Disk [Mounted]

Work [Mounted]

Workbench [Mounted]

Directories:

dh0	Work:DH0
MF	Work:TeX/METAFONT
TeX	Work:TeX/PasTeX
HELP	<LOCALE:Help>
LOCALE	Workbench:Locale
KEYMAPS	Workbench:Devs/Keymaps
PRINTERS	Workbench:Devs/Printers
REXX	Workbench:Rexxc
CLIPS	Ram Disk:Clipboards
T	Ram Disk:T
ENV	Ram Disk:ENV
ENVARC	Workbench:Prefs/Env-Archive
SYS	Workbench:
C	Workbench:C
S	Workbench:S
LIBS	Workbench:Libs + Workbench:Classes
DEVS	Workbench:Devs
FONTS	Workbench:Fonts
L	Workbench:L
TOOLS	[AmigaPowerTools:C]

Devices:

PIPE RAM CON RAW SER

PAR PRT HD0 DF0 CC0

HD1

1. Workbench:>

Die obige Liste zeigt, daß hier beispielsweise die Datei "Work:TeX/PasTeX-/ReadMe" auch mit "TeX:ReadMe" angesprochen werden kann. Das logische Geräte HELP: wurde mit dem Schlüsselwort DEFER und TOOLS: mit dem Schlüsselwort PATH den jeweiligen Verzeichnissen zugewiesen.

Siehe auch:

IF, ELSE, ENDIF, PATH, MOUNT

AUTOPOINT

Syntax:

2.0/2.1/3.0: AUTOPOINT [CX_PRIORITY <n>]

Schablone:

2.0/2.1/3.0: CX_PRIORITY/K/N

Pfad:

Extras:Tools/Commodities

Funktion:

Jedes Fenster wird automatisch aktiviert, wenn der Mauszeiger über das Fenster geschoben wird.

Beschreibung:

Bei diesem kleinen Hilfsprogramm scheiden sich die Geister, die einen schwören auf den kleinen Luxus (mann gönnt sich ja sonst nichts), die anderen könnten darauf gerne Verzicht. Entscheiden Sie selbst:

Wenn AUTOPOINT aktiviert ist, wird jedes Fenster automatisch aktiviert, in dem der Mauszeiger gerade steht. Es muß nicht mehr die linke Maustaste gedrückt werden. Gerade bei der Arbeit mit mehreren Shell-Fenstern kann dieses kleine Programm geradezu hinterlistige Charaktereigenschaften entwickeln.

Argumente:

CX_PRIORITY Mit der Priorität (im Verhältnis zu den anderen Commodities) können Sie festlegen, welches Programm die Informationen der Maus am frühesten bekommen soll.

Bemerkungen:

AUTOPOINT hat eine weitere Eigenschaft, die sehr unangenehm sein kann, wird das Programm wie ein normaler AmigaDOS-Befehl gestartet, blockiert es die Shell - bis es mit der Tastenkombination CTRL-E beendet wird. Es sollte daher immer als eigenständiger Prozess mit RUN gestartet werden -

wenn Sie wünschen am besten in der Datei "S:User-Startup". Das Programm läuft dann im Hintergrund und macht lediglich durch die automatische Fensteraktivierung auf sich aufmerksam.

Sie können AUTOPOINT auch wieder aus dem System entfernen, indem sie das Programm ein zweites Mal starten.

Noch ein kleiner Hinweis: Da es besser ist, die Commodities-Programme - sollen sie tatsächlich in eine Startup-Datei mit RUN aufgerufen werden - die Ausgabe auch gleich ins nichts umzuleiten (bsp.: "RUN AUTOPOINT >NIL:."), erscheint keine Fehlermeldung, falls die Befehlszeile nicht korrekt war. Wenn Sie wirklich Wert darauf legen, daß das Programm auch wirklich gestartet wurde (dies gilt für sämtliche Commoditiy-Programme), sollten Sie mit dem AmigaDOS-Befehl STATUS zumindest beim ersten Mal prüfen, ob das Programm korrekt gestartet werden konnte.

Siehe auch:

BLANKER, EXCHANGE, CLICKTOFRONT, FKEY, IHELP, NO-CAPSLOCK, RUN, STATUS

AVAIL

Syntax:

2.0/2.1/3.0: AVAIL [CHIP] [FAST] [TOTAL] [FLUSH]

Schablone:

2.0/2.1/3.0: CHIP/S,FAST/S,TOTAL/S,FLUSH/S

Pfad:

C:

Funktion:

Zeigt den derzeit verfügbaren RAM:-Speicher an.

Beschreibung:

Auf jedem Amiga ist der Speicher in zwei Typen unterteilt: das CHIP-RAM und das FAST-RAM. Das CHIP-RAM muß sich der Prozessor mit seinen Helfershelfern sprich den Coprozessoren teilen, während er auf das FAST-RAM alleine zugreifen kann, ohne aufpassen zu müssen, daß er andere stört. AVAIL zeigt nun jeweils die Kapazität von CHIP-, FAST-RAM, sowie beide zusammen ("total") an. Dabei wird nicht nur der noch verfügbare Platz angegeben, sondern auch, wieviel überhaupt im Amiga vorhanden sind ("Maximum"), wieviel davon derzeit gebraucht werden ("InUse"), wie groß das größte zusammenhängende Stück ("Largest") ist, und wieviel noch verfügbar sind ("Available"). Mit dem Schlüsselwort FLASH reagiert der Befehl jedoch etwas anders. Hier werden alle derzeit nicht benötigten Libraries, Devices und Schriftarten (Fonts) aus dem Speicher entfernt, um mehr Platz zu schaffen.

Argumente:

- CHIP/S Es werden nur die Daten über das CHIP-RAM angegeben.
- FAST/S Es werden nur die Daten über das FAST-RAM angegeben.
- TOTAL/S Die ist die normale Funktion, es werden alle Daten ausgegeben.
- FLUSH/S Hiermit werden die unnötigen Libraries und Devices aus dem Speicher entfernt.

Beispiele:

Die beiden folgenden Befehle wurden unmittelbar hintereinander ausgeführt. Hier wird deutlich, daß der Aufruf von AVAIL mit dem Schlüsselwort FLUSH einige Bytes an Speicher geschaffen hat.

1.Workbench:> AVAIL

Type	Available	In-Use	Maximum	Largest
chip	1230928	865200	2096128	1221392
fast	0	0	0	0

total 1230928 865200 2096128 1221392

1.Workbench:> AVAIL FLUSH

Type	Available	In-Use	Maximum	Largest
chip	1233672	862456	2096128	1221392
fast	0	0	0	0

total 1233672 862456 2096128 1221392

1.Workbench:>

Siehe auch: [PAG Sandini](#)

Info

A2024

Syntax:

2.1/3.0:A2024 [HBSTRT=<n>] [HBSTOP=<n>] [HSSTRT=<n>]
 [HSSTOP=<n>] [VBSTRT=<n>] [VBSTOP=<n>]
 [VSSTRT=<n>] [VSSTOP=<n>] [MINROW=<n>]
 [MINCOL=<n>] [TOTROWS=<n>] [TOTCLKS=<n>]
 [BEAMCON0=<n>]

2.0: Befehl nicht implementiert

Schablone:

2.1/3.0: HBSTRT/K,HBSTOP/K,HSSTRT/K,HSSTOP/K,
 VBSTRT/K,VBSTOP/K,VSSTRT/K,VSSTOP/K,MINROW/K,
 MINCOL/K,TOTROWS/K,TOTCLKS/K,BEAMCON0/K

2.0: Befehl nicht implementiert

Pfad:

DEVS:Monitors

Funktion:

Bei diesem Befehl handelt es sich um einen weiteren Monitortreiber, der nur bei Verwendung eines Commodore A2024-Monitors benutzt werden sollte.

Beschreibung:

Die Beschreibung des Befehls ADDMONTIOR gilt für A2024 sinngemäß, weshalb ich Sie auf den Befehl ADDMONITOR verweise - insbesondere beachten Sie bitte den Hinweis am Ende der Befehlsbeschreibung von ADDMONITOR.

Siehe auch:

ADDMONITOR

BINDDRIVERS

Syntax:

2.0/2.1/3.0: BINDDRIVERS

Schablone:

2.0/2.1/3.0: -

Pfad:

C:

Funktion:

Mit BINDDRIVERS werden Gerätetreiber den angeschlossenen Hard-ware-Erweiterungen zugeordnet.

Beschreibung:

Mit diesem Befehl, der in der original Datei "S:Startup-Sequence" schon integriert ist, werden Gerätetreiber für zusätzliche Hardwareerweiterungen geladen und gestartet. Die Konfiguration erfolgt automatisch über die "expansion.library". Der Befehl erwartet, daß die entsprechenden Gerätetreiber im Verzeichnis "SYS:Expansion" abgelegt sind und jeweils eine ".info"-Datei besitzen.

BINDMONITOR

Syntax:

2.0: BINDMONITOR [MONITORID] <ModeID> [MONITORNAME]
<ModeName>

2.1/3.0: Befehl nicht implementiert.

Schablone:

2.0: MONITORID/A,MONITORNAME/A

2.1/3.0: Befehl nicht implementiert.

Pfad:

SYS:System

Funktion:

PAG Sandini

Dieser Befehl weist einzelnen Bildschirmmodi Namen zu.

Beschreibung:

BINDMONITOR wird in erster Linie von der Systemsoftware benutzt, um den einzelnen Bildschirmmodi aussagekräftige Namen zu geben. Sie können somit Ihre eignen Bildschirmmodi-Bezeichnungen entwickeln.

Argumente:

MONITORID/A

Hier wird ein spezieller Code für die "graphics.library" eingeben, der sich auf einen bestimmten Bildschirmmodus bezieht. Es handelt sich um eine Hexadezimalzahl und muß (wie in der Programmiersprache C üblich) mit einem einführenden "0x" beginnen gefolgt von genau fünf Ziffern.

MONITORNAME/A

Hier können Sie den Namen des angegebenen Bildschirmmodus selbst bestimmen.

Bemerkungen:

Die "graphics.library" unterstützt beim Betriebssystem 2.0 folgende Modus-Nummern:

Nummer	Modus
0x08000	HiRes
0x08004	HiRes Interlaced
0x08020	SuperHires
0x08024	SuperHires Interlaced
0x19000	NTSC:Hires
0x19004	NTSC:Hires Interlaced
0x19020	NTSC:SuperHires
0x19024	NTSC:SuperHires Interlaced
0x29000	PAL:Hires
0x29004	PAL:Hires Interlaced
0x29020	PAL:SuperHires
0x29024	PAL:SuperHires Interlaced
0x39004	VGA-Lores
0x39005	VGA-Lores Interlaced
0x39024	Productivity
0x39025	Productivity Interlaced
0x41000	A2024_10Hz
0x49000	A2024_15Hz

BLANKER

Syntax:

- 2.0: BLANKER[SECONDS=<n>] [CX_POPKEY=<"hotkey(s)">]
[CX_POPUP=<YES/NO>] [CX_PRIORITY=<n>]
- 2.1/3.0: BLANKER [SECONDS=<n>] [CX_POPKEY=<"hotkey(s)">]
[CX_POPUP=<YES/NO>] [CX_PRIORITY=<n>] [CYCLECOLORS]
[ANIMATION]

Schablone:

- 2.0: SECONDS/K/N,CX_POPKEY/K,CX_POPUP/K,CX_PRIORITY/K/N
- 2.1/3.0: SECONDS/K/N,CX_POPKEY/K,CX_POPUP/K,CX_PRIORITY/K/N,
CYCLECOLORS/S,ANIMATION/S

Pfad:

Extras:Tools/Commodities

Funktion:

PAG Sandini

Schaltet bei längerer Wartetzeit den Bildschirm dunkel.

Beschreibung:

BLANKER ist wohl das wichtigste Programm der Commodities-Sammlung, vor allem, wenn Sie desöfteren den Amiga längere Zeit unbenutzt laufen lassen. Um zu verhindern daß das Computerbild die Bildschirmröhre durch den langen und vor allem konstanten Elektronenbeschuß "einbrennt" (v.a. bei Monochrommonitoren, aber auch bei Farbmonitoren), wird das Bild schwarz geschaltet, wenn der Amiga während einer bestimmten Zeit keine Anzeichen mehr erhält, daß der Anwender davor sitzt. Das ursprüngliche Bild ist geht dabei nicht verloren, es wird nur der dunkle Bildschirm darübergerlegt. Bewegt der Anwender dann seine Maus oder drückt eine beliebige Taste, so erscheint das ursprüngliche Bild in alter Frische. Auch das Commodity BLANKER sollte von der Shell mit RUN gestartet werden, da sonst die Shell lahm gelegt wird.

Argumente:

- SECONDS/K/N Die Zeit in Sekunden, nach der der Blanker in Aktion tritt. Voreingestellt sind hier 60 Sekunden, gültig sind alle Werte zwischen 1 und 9999.

- CX_POPKEY/K** Hier können Sie angeben, mit welcher Tastenkombination Sie das Fenster des BLANKER aufrufen wollen, um Änderungen vornehmen zu können. Voreingestellt ist die Kombination "CTRL-ALT-b".
- CX_POPUP/K** Hinter diesem Schlüsselwort muß entweder "YES" oder "NO" folgen. Wird hier YES angegeben (dieser Wert ist voreingestellt), so wird noch ein Fenster geöffnet, in dem Sie noch die Zeit und Art einstellen können, wann und wie BLANKER reagieren soll. Wird NO eingegeben, wird kein Fenster geöffnet, Blanker tritt sofort seinen Dienst an mit den Werten die voreingestellt sind, oder die als Argumente übergeben wurden.
- CX_PRIORITY/K** Hier können Sie festlegen, welche Priorität der BLANKER in Vergleich zu den anderen Commodities hat, Standard ist der Wert 0.
- ANIMATION/S** Wird dieses Schlüsselwort angegeben, erzeugt BLANKER Linienmuster, die über den Bildschirm tanzen.
- CYCLECOLORS/S** Wird dieses Schlüsselwort angegeben, so wechselt die Farbe der Linien ständig, oder wenn ANIMATION nicht angegeben wurde, wechseln die Bildschirmfarben direkt.

Beispiel:

Folgende Zeile startet den BLANKER automatisch mit der Priorität 2:

```
RUN BLANKER >NIL: CX_POPUP=NO CX_PRIORITY=2
```

Schreiben Sie diese Zeile in die Datei "S:User-Startup", und Sie brauchen sich keine Gedanken mehr zu machen, wenn der Amiga längere Zeit "unbeaufsichtigt" ist. Sollten Sie die neue Version des BLANKER mit Animation und colorcycling besitzen, gönnen Sie sich doch einmal diesen Augenschmauß.

Siehe auch:

AUTOPOINT, EXCHANGE, CLICKTOFRONT, FKEY, IHELP, NO-CAPSLOCK

BREAK

Syntax:

2.0/2.1/3.0: BREAK [PROCESS] <process> [ALL|C|D|E|F]

Schablone:

2.0/2.1/3.0: PROCESS/A,ALL/S,C/S,D/S,E/S,F/S

Pfad:

C:

Funktion:

Mit Break können Programme, die unter anderen Prozessen laufen, abgebrochen werden.

Beschreibung: PAG Sandini

Die meisten AmigaDOS-Befehle können mit der Tastenkombination Ctrl-C abgebrochen werden, wenn der Befehl nicht ganz ohne Fenster läuft. Wurde ein Befehl jedoch mit RUN gestartet, so reagiert dieser nicht auf "Ctrl-C", da er normalerweise kein eigenes Fenster besitzt. Doch mit den Befehl BREAK können auch diese Befehle gestoppt werden, indem man dem gewünschten Befehl eine Ctrl-C-Meldung schickt.

Es ist dabei unbedingt die Nummer des Prozesses anzugeben, unter dem der anzuhaltende Befehl läuft. Diese Nummer wird zum einen nach einem RUN-Start in eckigen Klammern angezeigt, kann aber auch mit dem Befehl STATUS (siehe auch dort) besorgt werden.

Argumente:

PROCESS/A Hier muß angegeben werden, welchen Befehl sie stoppen wollen. Erwartet wird die Nummer des Prozesses, unter dem der Befehl läuft.

ALL/S BREAK sendet dem Prozess alle folgenden vier Abbruchmeldungen.

- C/S** Dieser voreingestellte Wert ist das wichtigste Unterbrechungssignal. Bei einem erfolgreichen Programmabbruch wird im Shell-Fenster ein "***Break" angezeigt.
- D/S** Diese Unterbrechung wird zum Anhalten von Scriptdateien (wie etwa die Datei "S:User-Startup" u.ä.) verwendet.
- E/S** Auch dieser Schalter steht für ein eigenes Unterbrechungssignal, ein bestimmter Einsatzbereich dafür ist jedoch bislang noch nicht bekannt. Es gibt nur vereinzelt Programme, die dieses Abbruchsignal akzeptieren.
- F/S** Ctrl-F wird vor allem bei Programmen verwendet, für die ein eigenes Fenster vorgesehen ist. Dieses wird bei Ctrl-F bzw. dem entsprechenden BREAK-Befehl ggf. geöffnet und vor alle anderen Fenster platziert. Auch dieses Signal wird nicht von allen Programmen akzeptiert.

PAG Sandini

Beispiele:

Geben Sie in einem Shell-Fenster folgende Zeile ein:

```
1> dir
```

Sie erhalten den Inhalt des aktuellen Verzeichnisses und können die Ausgabe mit Ctrl-C unterbrechen, es erscheint "***Break". Geben Sie stattdessen die Zeile

```
3> RUN DIR
```

ein, so ist es nicht möglich, die Ausgabe mit Ctrl-C zu stoppen. Es erscheint beispielsweise:

```
[CLI 5]
```

```
3> _bullet_outlines (dir)
    courier (dir)
    diamond (dir)
    emerald (dir)
    garnet (dir)
    helvetica (dir)
```

opal (dir)	
ruby (dir)	
sapphire (dir)	
times (dir)	
topaz (dir)	
_bullet (dir)	
courier.font	diamond.font
emerald.font	garnet.font
helvetica.font	opal.font
ruby.font	sapphire.font
times.font	topaz.font

Wenn Sie jedoch in einem weiteren Shell-Fenster den Befehl

2> BREAK 5 C

eingeben, während die Ausgabe noch läuft, so wird diese Ausgabe mit "***Break" beendet. Natürlich können sich die Prozessnummern von denen im obigen Beispiel unterscheiden.

Siehe auch:

STATUS, RUN

BRU

Syntax:

2.0/2.1/3.0: BRU -<C|D|E|G|H|I|T|X> [Control options] [Selection options]
[files]

Schablone:

2.0/2.1/3.0: specify mode (-cdeg hitx)

Pfad:

Extras:Tools

Funktion:

Programm zum Erstellen von Sicherheitskopien der Festplatte.

Beschreibung:

Dieses von Fred Fish und der Firma EST Inc. entwickelte Programm erinnert sehr stark an die PD-Programme ZOO, ARC und LhARC, es ist nur wesentlich mächtiger - auch im Sinne von größer und speicherintensiver. Es gibt so viele Möglichkeiten, daß nur die wichtigsten in der Hilfestellung (nach Eingabe eines Fragezeichens als einziges Argument) Platz fanden. Auch hier können wir nur diese wichtigsten Argumente kurz behandeln.

Argumente:

- C Hiermit wird begonnen, ein Backup der Festplatte auf die Disketten im voreingestellten Laufwerk zu erstellen. Die Disketten werden dabei vorher automatisch formatiert, es müssen also keine vorformatierten Disketten eingelegt werden. Dies spart Zeit (wenn die gleichen Disketten immer wieder für ein Backup verwendet werden, oder wenn es sich um ein "PANIK-BACKUP" handelt, bei der man sowieso schon keine Nerven mehr und nach Murphys Gesetz sicherlich auch keine formatierten Disketten auf Lager hat.
- D Hier überprüft BRU nur, ob sich was zwischen dem letzten Backup und dem aktuellen Datenbestand geändert hat.

- E Hier gibt BRU nur an, wieviele Disketten benötigt werden, um ein Backup der kompletten Festplatte (oder einer Partition) zu erstellen.
- H Hier werden wesentlich ausführlicher die Argumente von BRU angezeigt. Hier empfiehlt es sich, die Ausgabe in eine Datei umzuleiten, damit der Text mit einem Anzeigeprogramm (wie More oder MultiView) oder einem beliebigen Texteditor in Ruhe gelesen werden kann. Beispiel:

Workbench:Tools.1> BRU >Bru.Help -h
- T Hier wird die Liste der in einem Archiv gespeicherten Dateien und Verzeichnisse angezeigt.
- X Mit dem Argument -X wird der Festplatteninhalt mit Hilfe der Backup-Disketten im voreingestellten Laufwerk wieder hergestellt.

PAG Sandini

Beispiel:

Wenn Sie weitere Informationen zu BRU haben wollen, geben Sie folgende Zeile ein:

```
1> BRU -H
```

Es werden anschließend alle Informationen zu BRU aufgezigt:

NAME

bru — backup and restore utility

SYNOPSIS

bru mode [control options] [selection options] [files]

MODE

- c create a new archive with specified files
- d find differences between archived files and current files
- e estimate media requirements for create mode

- g give only information from archive header
- h print this help information
- i inspect archive for consistency and data integrity
- t list archive table of contents for files
- x extract named files from archive

CONTROL OPTIONS

Sizes are specified in bytes. The scale factors 'M' or 'm', 'K' or 'k', or 'B' or 'b' can be appended to the size to indicate megabytes, kilobytes, or blocks (512 bytes) respectively.

- # str use debugging control string str
- a do not reset file access times after reads (now default)
- A flags Commodore Amiga specific flags:
 - c clear file archived bit after processing
 - f during filter mode reroute interactions to ipc port
 - i ignore file archived bit for selecting files
 - r reject files that have archived bit set
 - s set file archived bit after processing
- b N set archive buffer size to N bytes (scalable)
- B background mode, no interaction with operator
- C always chown extracted files to the user's uid/gid
- D on some systems, provides speedup via double buffering
- f file use specified file as archive ('-' for stdin/stdout)
- F fast mode, no checksum computations or checking
- I option an interaction option:
 - l,pathname write verbosity info to pathname
 - q,fifo write interaction queries to fifo
 - r,fifo read interaction replies from fifo
- L str label archive with given string (63 char max)
- l suppress warnings about unresolved links
- m limit directory expansions to same mounted filesystem
- N nbits use nbits for LZW compression (default 12); see -Z
- p pass over archive files by reading rather than seeking
- P opts special options for pathname handling and expansions
 - e turn off expansion of directories
 - E turn on expansion of directories

- f turn off filter mode (build internal file tree)
- F turn on filter mode (do not build internal tree)
- p turn off auto archiving of parent directory nodes
- P turn on auto archiving of parent directory nodes
- R exclude remotely mounted files for NFS/RFS systems
- s N specify size of archive media in bytes (scalable)
- S N turn on options to handle sparse files intelligently and set sparse file size threshold in bytes (scalable)
- v enable verbose mode (-vv and -vvv for more verbosity)
- w display action to be taken and wait for confirmation
- Z use LZW compression on archived files; see -N

FILE SELECTION OPTIONS

- n date select files modified since date
EX: 14-Apr-84,15:24:00
- o user select files owned by user, where user may be a symbolic user name, numeric user id, or file owned by user
- u abcdf use selected files in given class regardless of modification dates, where class is one or more of:
lpr
 - a use any file, same as giving all other args
 - b use block special files
 - c use character special files
 - d use directories
 - f use regular files (same as 'r')
 - l use symbolic links
 - p use fifos (named pipes)
 - r use regular files (same as 'f')

ENVIRONMENT VARIABLES

- BRUTAB full pathname of device table (EX: /etc/brutab)
- BRUTMPDIR full pathname of preferred temporary directory (EX: /tmp)
- SHELL preferred shell (EX: /bin/sh)

OTHER INFO

- release: 12.8
- variant: 1

bru id: Amiga Release 1.2
config: Mon Dec 31 15:07:01 MST 1990
archive: df0:
media size: 880k bytes usable
buffer size: 22k bytes
queries to: con:10/15/620/80/BRU Interaction Window
replies from: con:10/15/620/80/BRU Interaction Window
temporaries: ram:
device table: s:brutab
compression: 12 bits
copyright: Copyright (c) 1988, EST Inc. All Rights Reserved.

Bemerkung:

BRU (der Name ist eine Ankürzung für "Backup and Restore Utility") benötigt wesentlich mehr Stapelspeicher (Stack), als eine normale Shell hat. Sie können die Hilfsfunktion von BRU auch erst verwenden, wenn die Größe des Stapelspeichers mit dem Stack-Befehl auf mindestens 20000 gesetzt wurde, das Programm selbst verlangt aber nach eigenen Angaben mindestens 20480 Bytes Stack.

Siehe auch:

COPY, DISKCOPY, STACK

CALCULATOR

Syntax:

2.0: CALCULATOR [PUBSCREEN <screen>] [TAPE]

2.1/3.0: CALCULATOR [TAPE <device>] [PUBSCREEN <screen>]

Schablone:

2.0: PUBSCREEN, TAPE/K

2.1/3.0: TAPE/K, PUBSCREEN/K

Pfad:

Extras:Tools

Funktion:

Startet den integrierten Taschenrechner.

Beschreibung:

Mit diesem Befehl wird das Taschenrechnerprogramm aus der Tools-Schublade der Workbench gestartet. Es ist möglich die Rechnung wie bei einem Rechner mit Druckeinrichtung in eine Datei oder den Drucker umzuleiten.

Argumente:

TAPE/K Dieses Argument schafft die Möglichkeit, die Zwischenergebnisse der Rechnung in eine Datei oder ein anderes beliebiges physikalisches (Drucker, CON:-Fenster usw.) umzuleiten.

PUBSCREEN/K Öffnet das Fenster des Taschenrechner in einem sogenannten "Public Screen". Beachten Sie, daß ein Public Screen erst dann geschlossen werden kann, wenn alle Fenster - auch die von einem Fremdprogramm - geschlossen sind.

CD

Syntax:

2.0/2.1/3.0: CD [DIR] <directory|pattern>

Schablone:

2.0/2.1/3.0: DIR

Pfad:

in die Shell integriert

Funktion:

CD zeigt oder ändert das aktuelle Verzeichnis.

Beschreibung:

Wird der Befehl CD ohne Argumente angegeben, gibt er das aktuelle Verzeichnis aus. Wird jedoch ein Pfad zu einem Directory angegeben, so wird dieses Verzeichnis als neues aktuelles Verzeichnis erklärt, sofern es existiert. Wird das angegebene Verzeichnis nicht gefunden, erscheint die Fehlermeldung "Object not found".

CD unterstützt auch Namensmuster bzw. Jokerzeichen, sollte ein Muster jedoch für mehrere Verzeichnisse gültig sein, so wird ebenfalls eine Fehlermeldung angezeigt: "More than one directory matches".

Argumente:

DIR Pfadname oder Muster des neuen aktuellen Verzeichnisses.

/

Der sogenannte "Slash" wird zum einen als Trennzeichen zwischen den einzelnen Verzeichnisnamen verwendet. Steht er jedoch ein- oder mehrmals am Anfang des neuen Pfadnamen, so bewirkt er zum anderen, daß vom alten aktuellen Verzeichnis in der Hierarchie um die Anzahl der "/" nach oben gestiegen wird, bevor das neue aktuelle Verzeichnis von dort gesucht und deklariert wird.

: Wird der Pfadname von einem Doppelpunkt angeführt, so kehrt CD erst einmal in das Hauptverzeichnis des aktuellen Massenspeichers, und sucht von dort aus das neue aktuelle Verzeichnis.

Beispiele:

Mit der ersten Eingabe von CD wurde das aktuelle Verzeichnis neu festgelegt und mit dem zweiten CD nochmal angezeigt. Dies Beispiel ist hier zwar etwas unglücklich ausgewählt, da das aktuelle Verzeichnis schon in der Eingabeaufforderung gezeigt wird. Es kann jedoch vorkommen, daß ein Programm gestartet wird, das selbstständig das aktuelle Verzeichnis ändert, ohne daß das Prompt (die Eingabeaufforderung) geändert wird. In diesem Fall kann sich der Anwender oft wundern, warum sich den auf einmal der Inhalt dieses Verzeichnisses geändert hat. Wenn er jedoch den Befehl CD eingibt, so wird nicht nur das wirkliche aktuelle Verzeichnis angezeigt, auch die Eingabeaufforderung wird korrigiert:

```
1.Workbench:> cd devs/monitors
```

```
1.Workbench:Devs/Monitors> cd
```

```
Workbench:Devs/Monitors
```

Mit diesem Befehl wird zunächst in der Hierarchie eine Stufe nach oben gegangen, bevor das neue Verzeichnis "Printers" als aktuelles deklariert wird. Der zweite Befehl zeigt, daß ohne Eingabe des "/" ein Rücksprung ins ursprüngliche Verzeichnis nicht möglich ist.

```
1.Workbench:Devs/Monitors> cd /Printers
```

```
1.Workbench:Devs/Printers> cd monitors
```

```
object not found
```

Nun wird zunächst das nächsthöhere Verzeichnis als aktuelles festgelegt und versucht, nur mit einem Muster ein neues Verzeichnis als aktuelles zu deklarieren. Da auf das angegebene Muster mehr Directories passen, wird eine Fehlermeldung erzeugt:

```
1.Workbench:Devs/Printers> cd /
```

```
1.Workbench:Devs> cd #?i#?
```

```
More than one directory matches
```

Mit dem Doppelpunkt springt CD in Hauptverzeichnis des Datenträgers:

1.Workbench:Devs> cd :

1.Workbench:>

Bemerkungen:

Beachten Sie bitte, daß CD nur Verzeichnisnamen im Pfad akzeptiert, mit Dateinamen kann der Befehl nichts anfangen.

Seit Version 2.0 gibt es auch eine sehr komfortable Alternative, da AmigaDOS bzw. die Shell automatisch das aktuelle Verzeichnis wechselt wenn ein Pfadname eines bestimmten Verzeichnisses ohne irgendeinen Befehl eingegeben wird. Das Befehlswort "CD" kann also weggelassen werden. Beispiel:

1.Workbench:> Devs:Monitor

1.Workbench:Devs/Monitor> CD

Workbench:Devs/Monitor

1.Workbench:Devs/Monitor>

Siehe auch:

ASSIGN, PROMPT

CHANGETASKPRI

Syntax:

2.0/2.1/3.0: CHANGETASKPRI [PRI|PRIORITY] <priority> [PROCESS <process>]

Schablone:

2.0/2.1/3.0: PRI=PRIORITY/A/N,PROCESS/K/N

Pfad:

C:

Funktion:

CHANGETASKPRI ändert die Priorität eines beliebigen Prozesses.

Beschreibung:

Der Amiga ist für seine Multitasking-Fähigkeiten bekannt, diese dürfen aber nicht mit dem Multitasking des AmigaDOS gleichgestellt werden, wenngleich beide voneinander abhängen. Task und Process ist nicht das gleiche, diese Tatsache stiftet häufig Verwirrung.

Jedes Programm, daß von einer Shell oder auch von der Workbench gestartet wurde, ist ein "Process", das gegenüber den Tasks einige Privilegien genießt, die hier aber nicht erläutert werden können. Tasks sind niedrigere Systembestandteile (wie etwa der Maus-Treiber u.ä.), die für das AmigaDOS nicht manipulierbar sind. Der Befehl CHANGETASKPRI trägt einen falschen Namen, da er die Priorität von Prozessen ändert.

Jedem "Process" ist eine Priorität zugeordnet, die (theoretisch) im Bereich von -128 bis +127 liegen kann. Die Werte legen fest, wieviel Anteil der einzelne Prozess von der gesamten Rechenzeit für sich in Anspruch nehmen kann. Im Normalfall ist jedoch der Bereich der möglichen Priorität, die der Anwender festlegen kann, wesentlich kleiner. Der Anwender sollte nur Werte von -10 bis +10 verwenden (besser -5 bis +5), andernfalls kann er in Konflikt mit den anderen Systemprozessen und Tasks kommen, die sogar zum Absturz des Rechners führen können. Sie sollten daher diesen Befehl sehr umsichtig einsetzen.

QUALIFIER/K

Ab Version 2.1 kann hier eine Taste angegeben werden, die zusätzlich gedrückt werden muß, damit das gewünschte Fenster in den Vordergrund geholt wird. Es stehen folgende Tasten zur Auswahl:

Lalt, Left_Alt	Linke Alt-Taste
Ralt, Right_Alt	Rechte Alt-Taste
Ctrl, Control	Ctrl-Taste
None	keine Taste
a-Z	eine bestimmte Taste

Voreingestellt ist hier der Wert "NONE", wird CLICKTOFRONT jedoch von der Workbench über das Icon gestartet, so wird als Zusatzbedingung die linke Alt-Taste festgelegt (siehe "Information" aus dem Menü). Ein Buchstabe sollte nur in Verbindung mit anderen Tasten (Ctrl, Alt) als POPKEY definiert werden.

PAG Sandini

Siehe auch:

AUTOPOINT, BLANKER, EXCHANGE, FKEY, IHELP, NOCAPSLOCK

CLOCK

Syntax:

2.0: CLOCK [DIGITAL] [LEFT <n>] [TOP <n>] [WIDTH <n>]
[HEIGHT <n>] [24HOUR] [SECONDS] [DATE]

2.1/3.0: CLOCK [DIGITAL] [LEFT <n>] [TOP <n>] [WIDTH <n>]
[HEIGHT <n>] [24HOUR] [SECONDS] [DATE] [FORMAT <n>]
[PUBSCREEN <screen>]

Schablone:

2.0: DIGITAL/S,LEFT/N,TOP/N,WIDTH/N,HEIGHT/N,24HOUR/S,
SECONDS/S,DATE/S

2.1/3.0: DIGITAL/S,LEFT/N,TOP/N,WIDTH/N,HEIGHT/N,24HOUR/S,
SECONDS/S,DATE/S,FORMAT/S,PUBSCREEN/K

Pfad:

Workbench:Utilities

Funktion:

Es wird eine Uhr auf dem Bildschirm dargestellt.

Beschreibung:

Eine Uhr zu beschreiben, wird wohl nicht vom Autor gefordert, deshalb be-
fassen wir uns gleich mit den Argumenten.

Argumente:

DIGITAL/S Wenn Sie diesen Schalter übergeben, wird eine Digital-Uhr
dargestellt.

LEFT/N Die hier angegebene Zahl bestimmt den Abstand des Uhr-
Fensters vom linken Bildschirmrand.

TOP/N Analog zu LEFT wird hiermit der Abstand vom oberen
Bildschirmrand festgelegt.

PAG Sandini

AMIGA DOS 3.0

Directory "ram:" on Sunday 21-May-78

Umleitung.2	67361	—rwed Today	22:35:32
Umleitung.1	48595	—rwed Today	22:35:22
Umleitung	48595	—rwed Today	22:34:57
ENV	Dir	—rwed Today	22:30:26
Clipboards	Dir	—rwed Today	22:30:12
T	Dir	—rwed Today	22:30:23

3 files - 3 directories - 171 blocks used

Nobody is perfect - in der ersten Zeile wurde das Schlüsselwort "OPT" vergessen, so daß sich CMD doch nach der ersten Datei verabschiedete. Beim zweiten Versuch hat es jedoch geklappt. Nun sorgen wir dafür, daß CMD seinen Dienst beendet:

1.Work:DOS3.0> status

Process 1: Loaded as command: status

Process 2: Loaded as command: C:ConClip

Process 3: Loaded as command: hd0:m2emacs

Process 4: Loaded as command: sys:tools/commodities/blanker

Process 5: Loaded as command: cmd

1.Work:DOS3.0> break 5 c

Cmd redirection of parallel.device removed

1.Work:DOS3.0>

Das wars.

Siehe auch:

BREAK.

COLORS

Syntax:

2.0/2.1/3.0: COLORS [<bitplane> <screen type>]

Schablone:

2.0/2.1/3.0: BITPLANES, SCREENTYPE

Pfad:

Extras:Tools

Funktion:

COLORS ändert die Farben des Bildschirms im Vordergrund.

Beschreibung:

Wird der Befehl COLORS eingegeben, so erscheint ein Fenster, mit dem man die Farben des Bildschirms im Vordergrund ändern kann. Wird zusätzlich noch ein Auflösungs-Modus angegeben, erscheint das Fenster in einem eigenen Bildschirm mit der gewählten Auflösung.

Argumente:

BITPLANES Diese Zahl legt die Anzahl der verwendeten Bitplanes und damit der darstellbaren Farben fest. Es sind folgende Werte erlaubt:

Wert	Anzahl der Farben
1	2 Farben
2	4 Farben
3	8 Farben
4	16 Farben
5	32 Farben

SCREENTYPE Dieser Wert legt die Auflösung des Bildschirms für das COLORS-Fenster fest. Es sind folgende Zahlen möglich:

OFF/S Mit dem Schlüsselwort OFF wird kann CONCLIP inaktiviert werden, sodaß die Ausschneidefunktion nicht mehr verwendet werden kann. Normalerweise wird auch dieses Schlüsselwort nicht verwendet.

Bemerkungen:

Der Befehl CONCLIP löst sich selbständig von der Shell ab, von der er gestartet wurde, ein Start mit dem RUN-Befehl ist nicht nötig. Im Normalfall wird der Befehl jedoch automatisch von der "S:Startup-Sequence" gestartet.

PAG Sandini

COPY

Syntax:

2.0/2.1/3.0: COPY [FROM] {<name|pattern>} [TO] <name|pattern> [ALL]
[QUIET] [BUFIBUFFER <n>] [CLONE] [DATES] [NOPRO] [COM]
[NOREQ]

Schablone:

2.0/2.1/3.0: FROM/A/M,TO/A,ALL/S,QUIET/S,BUF=BUFFER/K/N,CLONE/S,
DATE/S,COM/S,NOPRO/S,NOREQ/S

Pfad:

C:

Funktion:

Mit COPY werden Dateien und Verzeichnisse kopiert.

Beschreibung:

COPY ist wohl einer der wichtigsten Befehle des AmigaDOS überhaupt und wird natürlich zum kopieren von Dateien und ganzen Verzeichnissen verwendet. Aber er kann wesentlich mehr, als nur einfach eine Datei von einem Ort zum anderen zu bringen. Die Hauptanwendung spiegelt sich in folgender Zeile:

1> COPY FROM quelledatei TO zielverzeichnis

Dieses Beispiel ist zwar nur Theorie, die Schlüsselwörter FROM und TO wurden angegeben, um die Eingaben noch eindeutiger zu machen, sie müssen nicht unbedingt verwendet werden. Gültig wäre genauso folgende Zeile,

1> COPY quelledatei zielverzeichnis

aber weniger einsichtig. Als Ziel kann sowohl ein physikalisches oder logisches Gerät als auch jedes sonstige Verzeichnis oder sogar der Drucker PRT: ein Konsolenfenster z.B. "CON:0/0/600/200/Ausgabefenster" angegeben werden.

Darüber hinaus kann gleichzeitig mit dem Kopieren einer Datei die neue Datei umbenannt werden, indem hinter dem Pfadnamen für das Zielverzeichnis der neue Name angehängt wird. Einzige Bedingung dafür ist, daß dieser Name nicht schon in diesem Verzeichnis vorkommt, andernfalls wird die schon vorhandene Datei mit den neuen Daten überschrieben oder wenn ein Verzeichnis den Namen trägt, wird die Datei unter dem alten Namen in diesem Unterverzeichnis abgelegt.

COPY ermöglicht das Kopieren von mehreren Dateien mit nur einem Befehl, indem die Dateien durch ein gemeinsames Namensmuster im FROM-Parameter ersetzt werden, oder die einzelnen Dateinamen durch Leerzeichen voneinander getrennt eingetippt werden, folgende Zeilen zeigen:

```
1.Work:DOS3.0> COPY FROM dos1 dos2 TO ram:Beispielverzeichnis  
ram:Beispielverzeichnis [created]
```

```
dos1..copied.
```

```
dos2..copied.
```

```
1.Work:DOS3.0>
```

PAG Sandini

Da in diesem Fall das Zielverzeichnis noch nicht existiert hat, wurde es von Copy erst geschaffen, anschließend wurden die Dateien "dos1" und "dos2" mit nur einem Copy-Befehl kopiert. Das gleiche Resultat hätte auch die Verwendung von Jokerzeichen geliefert:

```
1.Work:DOS3.0> COPY FROM dos#? TO ram:Beispielverzeichnis  
ram:Beispielverzeichnis [created]
```

```
dos1..copied.
```

```
dos2..copied.
```

```
1.Work:DOS3.0>
```

Zum Kopieren von Verzeichnissen sollte auch noch etwas erwähnt werden: Sie können bei der Quellangabe wie bei den Dateien auch den Namen eines Verzeichnisses angeben, dessen Inhalt komplett kopiert wird, im Unterschied zur Kopie bei Dateien, wird aber im Zielverzeichnis kein neues Unterverzeichnis mit dem Namen des alten Verzeichnisses angelegt. Wird beispielsweise mit dem Befehl

1> copy from t: to ram:

der Inhalt des logischen Gerätes T: ins Ram: kopiert, so werden die einzelnen Dateien von T: direkt ins Hauptverzeichnis von RAM: kopiert und kein Verzeichnis T: erzeugt. Soll im Zielverzeichnis wieder ein gleichnamiges Unterverzeichnis entstehen, so ist dies in den Pfadnamen des Zielverzeichnisses einzubinden:

1> copy fromt t: to ram:t

Hier wird auch im Ram ein Verzeichnis "T" erstellt. Alle in einem Verzeichnis enthaltenen Unterverzeichnisse werden jedoch in diesem Fall nicht kopiert. Es gibt aber die Möglichkeit, auch diese gleich mit zukopieren, indem an die Befehlszeile das Schlüsselwort "ALL" angehängt wird.

Argumente:

FROM/A/M

Der oder die Namen der Quelldateien oder -verzeichnisse, oder passende Namensmuster, die für die Dateien oder Verzeichnisse gelten, die kopiert werden sollen. Bei Eingabe eines Gerätes muß darauf geachtet werden, daß von diesem Gerät gelesen werden darf, von einem Drucker PRT: können keine Daten eingelesen werden, folglich kann dieses Gerät hier nicht angegeben werden.

TO/A

Hier muß der Pfadname des Verzeichnisses oder das Gerät angegeben werden, in das die Daten kopiert werden sollen. Es ist nicht möglich eine einzelne Datei in ein Verzeichnis zu kopieren, das erst geschaffen werden müßte. Fehlt nur eine Stufe in der Verzeichnishierarchie, so wird der letzte Name im Pfadnamen nicht als Verzeichnisnamen sondern als Dateiname interpretiert, die kopierte Datei erhält dann diesen neuen Namen. Fehlen mehrere Unterverzeichnisse in der Hierarchie, so wird der COPY-Befehl mit einer Fehlermeldung abgebrochen: "can't open xxx for output - object not found"

Beschreibung:

Der Befehl CPU ist für Amigas vorgesehen, die nicht den immer noch als Standard anzusehenden MC68000-Prozessor sondern höhere Typen ab dem MC68010 eingebaut haben. Vorher sollten noch ein paar Informationen zu den Prozessoren selbst genannt werden:

Überblick über die in Amigas verwendeten Prozessoren:

- 68000 Dieser Prozessor benutzt das sogenannte "one word prefetch", er holt sich schon die nächsten beiden Bytes des Programmes, während die beiden vorhergehende noch analysiert werden. Dies soll den Prozessor beschleunigen, kann aber unter bestimmten Bedingungen aber auch bremsen.
- 68010 Auch dieser Prozessor benutzt die Technik des "one word prefetch", hier wurde sie aber verbessert und zeigt ein ganz kleiner Speicherzugriffsschleifen deutliche Geschwindigkeitssteigerungen.
- 68020 Dieser Prozessor besitzt einen 256 Byte großen Befehlsspeicher, der auch bei größeren Schleifen ohne weitere Zugriffe auf das RAM auskommen kann, so daß die Geschwindigkeit beträchtlich gesteigert werden kann. Bei rein linearen Programmen kann aber auch diese Technik nichts beschleunigen.
- 68030 Hier kommen zu den 256 Befehlsspeicher noch einmal 256 Bytes Datenspeicher hinzu, die Befehlsbearbeitung entspricht der des 68020, wurde aber noch einmal beschleunigt (auch bei linearen Programmen).
- 68040 Diese Version ist technisch vergleichbar mit dem 68030, nur noch einmal schneller.

Die Prozessoren 68000, 68010 und 680EC20 (eine beschnittene Version des 68020) sind 16/32 Bit-Prozessoren, d.h. sie arbeiten intern mit einem 32-Bit breiten Bus, können aber auf den Speicher nur mit einem 16-Bit-Bus zugreifen. Die Typen 68020, 68030 und 68040 sind echte 32-Bit-Prozessoren, die auch über einen 32-Bit-Bus verfügen.

Wird der Befehl CPU ohne weitere Argumente benutzt, zeigt er einige Informationen zum System an:

1.Work:DOS3.0> CPU
System: 68020 (INST: Cache)

Diese Anzeige kann von Amiga zu Amiga unterschiedlich sein, je nachdem, welche Prozessoren unter welchen Bedingungen ihren Dienst leisten:

1.Work:DOS3.0> CPU
System: 68030 68881 FastROM (INST: Cache Burst)
(DATA: Cache NoBurst) 1.Work:DOS3.0>

Die erste Nummer zeigt den Typ des Hauptprozessors (CPU = Central Processing Unit), eine evtl. zweite Nummer gibt den Mathematischen Coprozessor an (diese sogenannte FPU oder "Floating Point Unit" ist nicht in jedem Amiga eingebaut). Als nächstes wird mit dem Wort "FastROM" gezeigt, daß das Kickstart-ROM in das RAM kopiert wurde, wodurch der Zugriff auf die Systemdaten (unter Speicherverlußt) beschleunigt wird. Um das ROM in das RAM zu kopieren, muß der Befehl CPU benutzt werden, allerdings können sich nur Systeme mit einer eingebauten MMU (Memory Management Unit) diesen Luxus leisten. Die folgenden Angaben in Klammern können auch sehr stark variieren: Die Möglichkeiten werden im folgenden aufgezeigt und anschließend näher erläutert:

INST: Cache	Instruction Cache (sprich: KäsCh) ist eingeschaltet.
INST: NoCache	Instruction Cache ist ausgeschaltet.
INST: Burst	Instruction Burst (sprich BörsT) ist eingeschaltet.
INST: NoBurst	Instruction Burst ist ausgeschaltet.
DATA: Cache	Data Cache ist eingeschaltet.
DATA: NoCache	Data Cache ist ausgeschaltet.
DATA: Burst	Data Burst ist eingeschaltet.
DATA: NoBurst	Data Burst ist ausgeschaltet.

Cache:

Die prozessor-eigenen Zwischenspeicher für Befehl (engl.: "Instruction") oder Daten (engl.: "Data") werden "Cache" genannt und besitzen üblicherweise eine Größe von je 256 Byte (sofern vorhanden, siehe obere Liste). Der Cache ist ein Bereich im RAM, der jedoch für alle anderen Coprozessoren Tabu ist (auch wenn dieser Bereich im CHIP-RAM liegt). Wenn der Prozessor hier Befehle ablegen kann, die er beispielsweise in einer Schleife immer wieder abzuarbeiten hat, so kann er seine Geschwindigkeit erheblich steigern, ähnliches gilt für den Data-Cache.

Burst:

Es handelt sich beim Burst um eine besondere Zugriffsart auf den Hauptspeicher, mit dessen Hilfe der Cache besonders schnell mit Daten bzw. Befehle gefüllt werden kann. Dadurch kann die Geschwindigkeit nochmals gesteigert werden. Diese Möglichkeit ist jedoch nur bei den Prozessoren 68030 und 68040 vorhanden und sollte nicht benutzt werden, wenn bestimmte für den Amiga 2000 entwickelte Erweiterungskarten verwendet werden.

Argumente:

CACHE/S	Alle Cache-Speicher werden verwendet
BURST/S	Der Burst-Modus für Daten und Befehle wird eingeschaltet.
NOCACHE/S	Sämtliche Cache-Speicher werden abgeschaltet.
NOBURST/S	Der Burst-Modus wird sowohl für Daten als auch für Befehle ausgeschaltet.
DATACACHE/S	Der Cache-Speicher wird nur für die Daten eingeschaltet.
DATABURST/S	Der Burst-Modus wird nur für die Daten eingeschaltet.
NODATACACHE/S	Der Cache-Speicher wird für die Daten abgeschaltet.
NODATABURST/S	Der Burst-Modus für die Daten wird abgeschaltet.

INSTCACHE/S	Der Cache-Speicher wird nur für die Befehle eingeschaltet.
INSTBURST/S	Der Burst-Modus wird nur für die Befehle eingeschaltet.
NOINSTCACHE/S	Der Cache-Speicher wird für die Befehle abgeschaltet.
NOINSTBURST/S	Der Burst-Modus für die Befehle wird abgeschaltet.
FASTROM/S	Sofern im Computer eine geeignete MMU (Memory Management Unit) vorhanden ist, werden die Daten vom ROM in das 32-Bit-RAM kopiert, um den Zugriff zu beschleunigen. Dieser Datenbereich wird anschließend schreibgeschützt, kann also nicht mehr geändert werden (dies ist ja auch im ROM der Fall).
NOFASTROM/S	Das FastROM wird abgeschaltet, der belegte Speicherplatz an das System zurückgegeben.
NOMMUTEST/S	Diese Option ist für Besitzer von Turbokarten gedacht, die durch einen Konstruktionsfehler abstürzen, wenn ein Programm prüft, ob im System bestimmte Coprozessoren vorhanden sind. Hierdurch wird die fehlerhafte Prüfung unterbunden. Dieser Parameter führt aber auch zum Rechnerabsturz, wenn er unter bestimmten Bedingungen (die mir nicht näher bekannt sind) verwendet wird.
TRAP/S	Diese Option sollte nur von Programmentwicklern verwendet werden (kann zum Rechnerabsturz führen)
NOTRAP/S	Diese Option sollte nur von Programmentwicklern verwendet werden (kann zum Rechnerabsturz führen).
COPYBACK/S	Dieser Parameter ist nur bei 68040-Prozessoren sinnvoll und schaltet den gleichnamigen Modus ein.

NOCOPYBACK/S	Dieser Parameter ist nur bei 68040-Prozessoren sinnvoll und schaltet den Modus "COPYBACK" aus.
EXTERNALCACHE/S	Dieser Parameter ist nur bei 68040-Prozessoren sinnvoll und aktiviert einen externen Cache-Speicher.
NOEXTERNALCACHE/S	Dieser Parameter ist nur bei 68040-Prozessoren sinnvoll und deaktiviert den externen Cache-Speicher.
CHECK/S	Dieses Schlüsselwort kann dazu benutzt werden um in Script- Dateien (ähnlich der "Startup-Sequence") das Vorhandensein bestimmte Prozessortypen zu überprüfen. Es gibt beispielsweise Programme, die in verschiedenen Versionen für die einzelnen Prozessortypen geliefert werden. Um beim Installieren auf eine Festplatte nur das Programm zu kopieren, daß an den eingebauten Prozessor angepaßt wurde, kann hiermit eine gezielte Installation möglich werden. Wenn die zu prüfende Einheit nicht vorhanden ist, wird ein "WARN" erzeugt (Fehlercode 5), der mit der IF-Anweisung abgefragt werden kann

Bemerkungen:

Es sind nicht alle Optionen bei den einzelnen Prozessortypen sinnvoll, in bestimmten Fällen kann ein unvorsichtiger Gebrauch des einen oder anderen Parameters zum Absturz des Rechners führen!

CROSSDOS

Syntax:

2.0: Befehl nicht implementiert.

2.1/3.0: CROSSDOS [CX_PRIORITY <priority>] [CX_POPKEY <key>]
[CX_POPUP <yes/no>]

Schablone:

2.0: Befehl nicht implementiert.

2.1/3.0: CX_PRIORITY/K/N,CX_POPKEY/K,CX_POPUP/K

Pfad:

Extras:Tools/Commodities

PAG Sandini

Funktion:

Können auf einem Amiga-Diskettenlaufwerk MS-DOS-Disketten gelesen bzw. geschrieben werden, so können hiermit Textfilter und Konvertierungsmöglichkeiten ausgewählt werden.

Beschreibung:

CrossDOS ist zum einen ein Überbegriff für alle Programme, die es in ihrer Gesamtheit dem Amiga ermöglichen, mit MS-DOS-Disketten (im 720-KByte-Format) zu arbeiten. Der Befehl CrossDOS trägt dazu seinen Teil bei, indem hiermit bestimmte Textkonvertierungsmöglichkeiten und Textfilter ausgewählt werden können, mit denen dann die Texte auf der MS-DOS-Diskette den Amiga-Anforderungen angepaßt werden. Der Befehl CROSSDOS kann nur dann sinnvoll eingesetzt werden, wenn einem Diskettenlaufwerk bereits ein MS-DOS-Handler zugewiesen wurde, d.h. das Device "PCx:" aus dem Verzeichnis "Devs:DosDrivers" bzw. "Storage/DosDrivers" aktiviert wurde.

Eine detailliertere Beschreibung finden Sie im Kapitel zur Workbench.

Argumente:

CX_PRIORITY/K/N

Die Priorität, mit der dieses Programm gegenüber den anderen Commodity-Programmen eingestuft wird.

CX_POPKEY/K

Hier kann festgelegt werden, mit welchen Tastenkombinationen das Fenster von CROSS DOS aufgerufen werden kann. Zusätzlich zu den beim Befehl CLICKTOFRONT aufgeführten Möglichkeiten bietet CROSSDOS noch die besondere "Taste" "DISKINSERTED". Hier erscheint jedesmal das Fenster, wenn eine Diskette in ein Laufwerk gelegt wird.

CX_POPUP/K

Mit einem einfachen "yes" oder "no" bestimmen Sie, ob auch beim Aufruf von CROSSDOS das Fenster erscheinen soll oder nicht.

Siehe auch:

AUTOPOINT, BLANKER, CLICKTOFRONT, EXCHANGE, FKEY, IHELP, NOCAPSLOCK, Kapitel "CrossDOS".

PAG Sandini

DATE

Syntax:

2.0/2.1/3.0: DATE [[DAY] <day>] [[DATE] <date>] [[TIME] <time>]
[TO|VER <filename>]

Schablone:

2.0/2.1/3.0: DAY,DATE,TIME,TO=VER/K/A

Pfad:

C Funktion: Zeigt oder ändert Systemdatum bzw. Systemzeit.

Beschreibung:

Wird DATE ohne Argumente eingeben, so wird das aktuelle Datum und die Uhrzeit im 24-Stunden-Format angezeigt.

Grundsätzlich hängen die gültigen Eingaben beim Tag und dem Datum davon ab, welche Sprache mit dem Preferences-Programm "Locale" eingestellt wurde. Da wir aber davon ausgehen können, daß entweder Deutsch oder Englisch eingestellt ist, werden alle Beispiele und Argumente bei Englisch normal und bei Deutsch in Klammern gesetzt. Mit DAY können Sie einen Wochentag eingeben, es werden sowohl relative Tage wie "yesterday" ("gestern"), "today" ("heute") oder "tomorrow" ("morgen") als auch alle Wochentage von "Monday" ("Montag") bis "Sunday" ("Sonntag") akzeptiert. Es wird das Datum in diesem Fall immer auf den nächsten angegebenen Wochentag gesetzt, nur mit "yesterday" ("gestern") kann das Datum um einen Tag zurückgesetzt werden.

Auch die Datumsangabe hängt von der eingestellten Sprache ab. Es gibt zwei Möglichkeiten, das Datum anzugeben: Entweder im Format DD-MMM-YY (Tag-Monat-Jahr), bei der für MMM die ersten drei Buchstaben des Monats eingesetzt werden müssen oder im Format DD-MM-YY, bei der der Monat als zweistellige Zahl eingeben wird.

Die Uhrzeit ist unabhängig von der bevorzugten Sprache im Format HH:MM:SS (je zweistellig Stunde:Minuten:Sekunden) einzugeben, die Angabe der Sekunden kann auch entfallen.

Argumente:

DAY	Name eines Wochentages oder relative Angabe in der jeweils eingestellten Sprache.
DATE	Datum im Format DD-MMM-YY oder DD-MM-YY. Wird der Monat mit seinen drei Anfangbuchstaben eingegeben, so ist auf die Schreibweise in der eingestellte Sprache zu achten. Er kann aber auch als zweistellige Zahl angegeben werden.
TIME	Hier wird die Uhrzeit im Format HH:MM oder HH:MM:SS angegeben.
TO=VER/K/A	Wird hier ein Dateiname angegeben, so wird die Zeitangabe in die angegebene Datei umgeleitet. Vorsicht: Eine eventuell schon vorhandene Datei gleichen Namens wird überschrieben.

Beispiele:

PAG Sandini

Sie wollen wissen, wie spät es ist, bitte sehr:

```
1> DATE Monday 11-Apr-94 16:27:01
```

Aber der Tag war gestern, also muß es korrigiert werden:

```
1.Work:DOS3.0> date tomorrow
```

```
1.Work:DOS3.0> date Tuesday 12-Apr-94 16:27:11
```

Und jetzt brauchen wir ein Alibi für den nächsten Samstag, alle ab jetzt vorgenommenen Manipulationen werden auf den folgenden Samstag datiert.

```
1.Work:DOS3.0> date saturday
```

```
1.Work:DOS3.0> date Saturday 16-Apr-94 16:32:07
```

Wenn Sie wieder auf den Montag zurück stellen wollen, kann dies nicht über den Wochentag geschehen, denn hier wird der folgende Montag gesetzt:

```
1.Work:DOS3.0> date monday
```

```
1.Work:DOS3.0> date Monday 18-Apr-94 16:32:12
```

Und bei der Wiederholung der Montag in zwei Wochen:

```
1.Work:DOS3.0> date monday
```

```
1.Work:DOS3.0> date Monday 25-Apr-94 16:32:18
```

Bemerkungen:

Viele Amigas besitzen zwei verschiedene Uhren, neben der softwaremäßigen auch eine akkugepufferte Hardware-Uhr, die das Datum und die Zeit auch beibehält, wenn der Amiga abgeschaltet ist. Der Befehl DATE ändert nur die softwareseitige Uhr, um das eingestellte Datum und die Zeit in die akkugepufferte Uhr zu übernehmen, muß nach der Einstellung mit DATE der Befehl SETCLOCK verwendet werden.

Das frühest mögliche Datum des Amigas ist der 1. Januar 1978

Siehe auch:

SETCLOCK

PAG Sandini

DBLNTSC

Syntax:

2.0/2.1: Befehl nicht implementiert

3.0: DBLNTSC [HBSTRT=<n>] [HBSTOP=<n>] [HSSTRT=<n>]
[HSSTOP=<n>] [VBSTRT=<n>] [VBSTOP=<n>]
[VSSTRT=<n>] [VSSTOP=<n>] [MINROW=<n>]
[MINCOL=<n>] [TOTROWS=<n>] [TOTCLKS=<n>]
[BEAMCON0=<n>]

Schablone:

2.0/2.1: Befehl nicht implementiert

3.0: HBSTRT/K, HBSTOP/K, HSSTRT/K, HSSTOP/K, VBSTRT/K, VBSTOP/K,
VSSTRT/K, VSSTOP/K, MINROW/K, MINCOL/K, TOTROWS/K,
TOTCLKS/K, BEAMCON0/K

Pfad:

Devs:Monitors

Funktion:

Ändert den Bildschirmmodus und die Auflösung:

Argumente:

Eine detaillierte Beschreibung der Argumente finden Sie bei dem Befehl ADDMONITORS.

Verwenden Sie diesen Befehl nur zur Erweiterung der Monitor-Liste. Ein unsachgemäßer Gebrauch kann zur Zerstörung des angeschlossenen Monitors oder Fernsehers führen.

Siehe auch:

ADDMONITORS

DBLPAL

Syntax:

2.0/2.1: Befehl nicht implementiert

3.0: DBLPAL [HBSTRT=<n>] [HBSTOP=<n>] [HSSTRT=<n>]
[HSSTOP=<n>] [VBSTRT=<n>] [VBSTOP=<n>]
[VSSTRT=<n>] [VSSTOP=<n>] [MINROW=<n>]
[MINCOL=<n>] [TOTROWS=<n>] [TOTCLKS=<n>]
[BEAMCON0=<n>]

Schablone:

2.0/2.1: Befehl nicht implementiert

3.0: HBSTRT/K,HBSTOP/K,HSSTRT/K,HSSTOP/K,VBSTRT/K,VBSTOP/K,
VSSTRT/K,VSSTOP/K,MINROW/K, MINCOL/K,TOTROWS/K,
TOTCLKS/K,BEAMCON0/K

Pfad:

Devs:Monitors

Funktion:

Ändert den Bildschirmmodus und die Auflösung:

Argumente:

Eine detaillierte Beschreibung der Argumente finden Sie bei dem Befehl ADDMONITORS.

Verwenden Sie diesen Befehl nur zur Erweiterung der Monitor-Liste. Ein unsachgemäßer Gebrauch kann zur Zerstörung des angeschlossenen Monitors oder Fernsehers führen.

Siehe auch:

ADDMONITORS

DELETE

Syntax:

2.0/2.1/3.0: DELETE [FILE] {<name|pattern>} [ALL] [Q|QUIET] [FORCE]

Schablone:

2.0/2.1/3.0: FILE/M/A,ALL/S,Q=QUIET/S,FORCE/S

Pfad:

C:

Funktion:

Delete löscht Dateien oder Verzeichnisse.

Beschreibung:

Der Befehl DELETE benötigt immer die Namen der Dateien oder Verzeichnisse, welche gelöscht werden sollen. Es sind auch Namensmuster (siehe Kapitel "Jokerzeichen") erlaubt, doch sollten diese mit größter Vorsicht verwendet werden, da oftmals ein Muster auf verschiedene Dateien paßt, so daß schnell ungewollt Dateien gelöscht werden können. Es werden immer alle Dateien gelöscht, deren Namen dem angegebenen Muster gehorcht.

Wird auf die Angabe von Jokerzeichen verzichtet, ist DELETE in der Lage in einem Durchgang bis zu zehn explizit angegebene Dateien zu löschen. Wenn sich die Dateien nicht im aktuellen Verzeichnis befinden, so muß zusätzlich der komplette Pfad angegeben werden. Unter normalen Bedingungen ist es nicht möglich Unterverzeichnisse zu löschen, die noch Dateien enthalten. Hierzu dient dann die Option "ALL".

ACHTUNG: Vor dem Löschvorgang erfolgt keine Bestätigungsrückfrage. Ein Fehler in der Namensmuster-Liste kann schwere Folgen nach sich ziehen, da auch Dateien gelöscht werden können, die nicht gelöscht werden sollten. Es ist nicht möglich, einmal gelöschte Daten wieder herzustellen.

Argumente:

- ALL/S** Normalerweise werden nicht leere Unterverzeichnisse nicht gelöscht. Wird jedoch diese Option verwendet, so werden erst alle Dateien in den vorhandenen Unterverzeichnis gelöscht, bevor es selbst entfernt wird. Dieser Schalter sollte nur mit größter Vorsicht verwendet werden.
- Q=QUIET/S** Dieses Schlüsselwort unterdrückt die Meldungen zum Arbeitsfortgang des Befehls DELETE. Es erscheinen nur Fehlermeldungen - etwa weil ein zu löschendes Verzeichnis noch Dateien enthält.
- FORCE/S** Dateien können über die Schutzbits (siehe Befehl PROTECT) vor dem Löschen "geschützt" werden. Doch mit dem Schlüsselwort FORCE wird DELETE gezwungen, auch so geschützte Dateien zu entfernen.

Bemerkungen: PAG Sandini

Verwenden Sie Namensmuster nur mit äußerster Vorsicht. Es ist möglich, vorher die Wirkung des vorgesehenen Musters zu Testen, indem der Befehl DIR <Muster> verwendet wird. Es werden nur die Dateien angezeigt, deren Namen dem Muster genügen.

Ein Verzeichnis, daß von irgend einer Shell als aktuelles Verzeichnis erklärt wurde, oder auf das ein logisches Gerät zeigt, kann auch dann nicht gelöscht werden, wenn es vollkommen leer ist. In diesem Fall müssen erst alle Definitionen von logischen Geräten und von aktuellen Verzeichnissen geändert werden, bevor dieses Verzeichnis gelöscht werden kann.

Siehe auch:

DIR, PROTECT

DIR

Syntax:

2.0/2.1/3.0: DIR [[DIR] <dir|pattern>] [OPT <A|I|D|F|AF>] [ALL]
[DIRS] [FILES] [INTER]

Schablone:

2.0/2.1/3.0: DIR,OPT/K,ALL/S,DIR/S,FILES/S,INTER/S

Pfad:

C:

Funktion:

Der Inhalt von Verzeichnissen wird alphabetisch angezeigt.

Beschreibung:

RAG Sandini

Der Befehl DIR wird von allen AmigaDOS-Befehlen wohl am häufigsten benutzt, weil es natürlich sehr wichtig ist, den Inhalt von bestimmten Verzeichnissen zu wissen. Um so erstaunlicher ist, daß er im Verzeichnis "C" geblieben ist, und nicht wie viele seiner Artgenossen in die Shell integriert wurde. (Man kann nur vermuten, daß er so leichter erweitert und verbessert werden kann.) Man wird aber dem Befehl DIR nicht gerecht, wenn man sagt, daß er nur den Inhalt von Verzeichnissen, logischen Geräten oder Disketten zeigen kann. Es ist auch möglich, den Inhalt bestimmter Textdateien anzuzeigen (ähnlich dem Befehl TYPE).

Wird DIR ohne weitere Parameter verwendet, so wird der Inhalt des aktuellen Verzeichnisses angezeigt. Es können aber Pfadnamen oder Namensmuster von anderen Verzeichnissen angegeben werden, deren Inhalt dann gezeigt wird. Es kann in Script-Dateien (ähnlich der "Startup-Sequence") aber zu Problemen führen, wenn ein Namensmuster zu keinem der Einträge im angegebenen Verzeichnis paßt, denn dann erzeugt DIR eine Fehlermeldung (bei AmigaDOS 2.0 Fehler "232: no more entries", ab Version 2.1 "205: object not found") und einen Returncode 20, der unter Umständen zum Abbruch der Befehlsbearbeitung führen kann.

Argumente:

- DIR** Der Name des Verzeichnisses, dessen Inhalt in alphabetischer Reihenfolge angezeigt werden sollen, oder ein Namensmuster von Dateien in einem Verzeichnis, die alphabetisch sortiert dargestellt werden sollen.
- ALL/S** Wird dieses Schlüsselwort gesetzt, so werden neben dem Inhalt des eigentlichen Verzeichnisses auch die Inhalte sämtlicher Unterverzeichnisse angezeigt. Dabei wird die Ausgabe durch ein immer größeres Einrücken der Namen so geschachtelt, wie die Verzeichnisse in der Hierarchie vorliegen. Je tiefer sich ein Verzeichnis oder eine Datei in der Struktur befindet, um so weiter wird die Zeile eingerückt. Wird dieses Schlüsselwort jedoch in Verbindung mit Jokerzeichen verwendet, so werden nur das aktuelle Verzeichnis und alle direkten Unterverzeichnisse gefunden.
- DIR/S** Dieser Schalter bewirkt, daß nur die in einem Verzeichnis vorhandenen Unterverzeichnisse angegeben werden, die Dateien werden nicht ausgegeben.
- FILES/S** Hier werden nur die Dateien angezeigt, die Verzeichnisse werden übergangen.
- INTER/S** Dieses Schlüsselwort kann oft sehr hilfreich sein - bisweilen aber auch etwas verwirrend. Dabei wird DIR in einen interaktiven Modus geschaltet, in dem DIR viele weitere Möglichkeiten bietet, etwa zum anzeigen von Texten oder zum löschen ganzer Dateien.
- Es werden der Reihe nach alle Dateien einzeln gefolgt von einem Fragezeichen angezeigt, anschließend erwartet DIR eine Reaktion des Anwenders. Es können dabei folgende Antworten gegeben werden:

E bzw. ENTER Diese Antwort ist nur bei Verzeichnissen möglich, die hiermit betreten werden (engl: Enter) und dessen Inhalt ebenso untersucht werden kann. Enthält dieses Verzeichnis weitere Unterverzeichnisse, können diese wiederum mit "E" besucht werden. Verzeichnisse werden durch die Eingabeaufforderung "(dir)?" gekennzeichnet.

B bzw. BACK B macht die Eingabe "E" vorzeitig rückgängig, springt also wieder eine Stufe zurück (engl: Back). Befindet sich DIR schon im obersten Verzeichnis, so wird der Befehl durch "B" abgebrochen.

DEL o. DELETE Eine Datei oder ein leeres Verzeichnis wird gelöscht. "DEL" bezeichnet hierbei nicht die gleichnamige Taste, sondern es sind die drei Buchstaben als Akürzung oder DELETE selbst einzugeben.

C bzw. COM Seit der Version 1.3 gibt es die Möglichkeit nach dem Buchstaben "C" einen anderen AmigaDOS-Befehl einzuschieben, bevor mit dem interaktiven DIR-Befehl fortgefahren wird. Dieser Befehl kann direkt hinter "C" eingegeben werden, wie er nach einer normalen Shell-Eingabeaufforderung eingetippt werden würde. Es kann aber auch der Buchstabe "C" allein eingegeben werden, dann bringt DIR eine eigene Befehlseingabeaufforderung. Beispiel:

1.Work:DOS3.0> DIR INTER

Bilder (dir) ? e

Workbench (dir) ? e

Blanker3.0 ? c

Command ? info

Mounted disks:

Unit	Size	Used	Free	Full	Errs	Status	Name
RAM:	39K	39	0	100%	0	Read/Write	Ram Disk
HD0:	8267K	7448	9086	45%	0	Read/Write	Workbench
DF0:	No disk present						
HD1:	107M	151876	68706	69%	0	Read/Write	Work

Volumes available:

Ram Disk [Mounted]

Work [Mounted]

Workbench [Mounted]

Blanker3.0 ? b

System-Verzeichnisse (dir) ? ?

E=ENTER/S,B=BACK/S,DEL=DELETE/S,Q=QUIT/S,C=COM/S,COMMAND: q

1.Work:DOS3.0>

NOG Sandini

Noch drei wichtige Hinweise, die mit der Option
“C” des interaktiven Modus von DIR eng verknüpft sind:

1. Verwenden Sie hier nie den Befehl FORMAT, da hier die Diskette gelöscht wird, während ein anderer noch laufender Befehl (DIR ist in diesem Fall nur unterbrochen und wird später fortgesetzt) die gleiche Diskette lesen möchte. Dies wäre, als ob jemand Zeitung lesen würde die ein anderer gerade anzündet.
2. Starten Sie keinen weiteren interaktiven DIR-Befehl, da dann die Ausgaben undurchschaubar werden. Man kann ja auch nicht mit einem Fernseher gleichzeitig zwei verschiedene Programme anschauen.
3. Wenn Sie dennoch einen weiteren interaktiven DIR-Befehl unbedingt benötigen, können Sie mit der Zeile “C NEWSHELL” eine neue Shell öffnen, in dem ein weiterer DIR-Befehl problemlos interaktiv laufen kann.

Q bzw.
QUIT

Mit "Q" oder "QUIT" kann der DIR-Befehl an jeder beliebigen Stelle abgebrochen werden. Eine Unterbrechung mit Ctrl-C oder dem Befehl "BREAK C" ist im interaktiven Modus von DIR nach meinen Erfahrungen nicht möglich.

T bzw.
TYPE

Mit diesem Parameter kann der Inhalt der gerade angezeigten Datei - der Parameter ist nur bei Dateien erlaubt - wie bei dem gleichnamigen TYPE-Befehl angezeigt werden. Allerdings darf man keine Sonderwünsche haben, eine Anzeige der Datei in HEX-Code oder mit durchnummerierten Zeilen, wie beim richtigen TYPE-Befehl darf man nicht verlangen. Hier kann jedoch der Parameter "C" sinnvoll eingesetzt werden.

?

Hier kann sich jeder eine kurze Hilfe holen, welche Möglichkeiten gerade bei der angezeigten Datei oder dem angegebenen Verzeichnis vorhanden sind. Es werden die gleichen Schlüsselwörter verwendet, wie in dieser Aufzählung.

Beispiele:

Wenn Sie den Inhalt des logischen Gerätes DEVS: ansehen wollen, können Sie folgenden Befehl eingeben:

1.Work:DOS3.0> DIR DEVS:

Monitors (dir)

DataTypes (dir)

DOSDrivers (dir)

Printers (dir)

Keymaps (dir)

clipboard.device DataTypes.info

DOSDrivers.info	Keymaps.info
mfm.device	Monitors.info
parallel.device	postscript_init.ps
printer.device	Printers.info
serial.device	system-configuration

Und wenn Ihnen die Dateien alleine genügen:

1.Work:DOS3.0> DIR DEVS: FILES

clipboard.device	DataTypes.info
DOSDrivers.info	Keymaps.info
mfm.device	Monitors.info
parallel.device	postscript_init.ps
printer.device	Printers.info
serial.device	system-configuration

1.Work:DOS3.0>

Siehe auch:

CD, LIST, TYPE

PAG Sandini

DISKCHANGE

Syntax:

2.0/2.1/3.0: DISKCHANGE [DRIVE] <device>:

Schablone:

2.0/2.1/3.0: DRIVE/A

Pfad:

C:

Funktion:

Wenn in einem 5,25"-Laufwerk eine Diskette gewechselt wurde, kann mit DISKCHANGE der Amiga davon informiert werden.

Beschreibung:

Dieser Befehl ist nur bei Arbeiten mit einem 5,25-Zoll-Laufwerk oder anderen Laufwerken für Speichermedien ohne Wechselerkennung (beispielsweise Streamer) notwendig, da hier der Amiga nicht selbstständig einen Diskettenwechsel erkennen kann. Bei 3.5"-Diskettenlaufwerken ist die Erkennung von Diskettenwechseln normalerweise automatisiert.

Argument:

DRIVE/A Dieser Befehl benötigt nur den Namen des Gerätes, dessen Massenspeicher ausgetauscht wurde.

Bemerkungen:

Der Befehl kann auch verwendet werden, um dem Betriebssystem mitzuteilen, daß der Namen der gerade einliegenden Diskette soeben mit dem Befehl RELABEL geändert wurde. Normalerweise wird diese Änderung nicht automatisch von Befehlen wie INFO erkannt. Eine Verwendung von DISKCHANGE kann hier das Herausnehmen und Neueinlegen der Diskette überflüssig machen.

Siehe auch:

RELABEL, INFO

DISKCOPY

Syntax:

2.0/2.1/3.0: DISKCOPY [FROM] <drive>: [TO] <drive>: [NAME <name>]
[NOVERIFY] [MULTI]

Schablone:

2.0/2.1/3.0: FROM/A,TO/A,NAME/K,NOVERIFY/S,MULTI/S

Pfad:

SYS:System

Funktion:

Eine Diskette wird kopiert.

Beschreibung:

Der Befehl DISKCOPY macht genau das, was man von ihm erwartet: er kopiert Disketten. In der Praxis wird dem Befehl aber nicht gebührend Aufmerksamkeit geschenkt, ist er doch eigentlich einer der grundlegendsten Befehle des AmigaDOS überhaupt. Es ist ein Ding der Unmöglichkeit, die Notwendigkeit dieses Befehles zu beschreiben, denn ohne DISKCOPY kann der Anwender in ganz knifflige Situationen kommen, beispielsweise wenn die Workbench-Diskette ihren Dienst versagt. Der Benutzer wäre dann zum Spielen verdonnert.

Nachdem der Befehl eingegeben wurde, wird der Anwender aufgefordert, die Quell- und Ziel-Diskette in die Laufwerke einzulegen und anschließend die RETURN-Taste zu drücken. Vergewissern Sie sich, daß die richtige Diskette im richtigen Laufwerk einliegt (mit den Angaben des Befehls vergleichen!).

Argumente:

FROM/A Wird der Name des Diskettenlaufwerkes angegeben, in dem die(schreibgeschützte) Quelldiskette liegt.

TO/A Hier wird der Name des Diskettenlaufwerkes eingegeben, in dem die Zieldiskette liegt. Ist nur ein Laufwerk vorhanden,

so sind sowohl bei FROM als auch bei TO "df0:" anzugeben. Aber keine Angst, der Amiga sagt Ihnen schon bei Bedarf, wann die Disketten gewechselt werden sollen.

NAME/K Normalerweise erhält die Kopie den gleichen Namen wie die Originaldiskette. Wird jedoch ein anderer Namen gewünscht, kann der zusammen mit dem Schlüsselwort "NAME" angegeben werden.

NOVERIFY/S Dieses Schlüsselwort sorgt dafür, daß die normalerweise übliche Prüfung der Zieldiskette auf Fehler nicht durchgeführt wird. Die Kopie einer Diskette wird dadurch beschleunigt.

MULTI/S Mit dieser Option werden die Daten der Quelldiskette - wenn möglich - vollständig in den Speicher geladen. Es können dann nacheinander mehrere Kopien der Originaldiskette erstellt werden, ohne daß jedesmal die Originaldiskette wieder eingelegt werden muß.

PAG Sandini

Beispiel:

Um die Diskette von Laufwerk "df1:" auf die Diskette "df0:" zu kopieren geben Sie folgende Zeile ein:

```
1> DISKCOPY FROM df1: TO df0:
```

Amiga-Anwender mit nur einem Laufwerk benutzen den Befehl wie folgt:

```
1> DISKCOPY df0: df0:
```

Bemerkungen:

Der Befehl DISKCOPY kann auch verwendet werden, um die resetfeste RAM-Disk "RAD:" zu kopieren.

Beachten Sie, daß die Originaldiskette vor dem Kopiervorgang schreibgeschützt ist, sonst hat eine Verwechslung der Laufwerke verheerende Folgen. In diesem Fall wird sonst die Originaldiskette mit den Daten der eigentlich zu beschreibenden Diskette beschrieben.

Während des Kopiervorgangs zeigt DISKCOPY, wieviel Spuren er gelesen, bzw. geschrieben hat ("reading ##,") und wieviel noch kopiert werden müssen ("## to go").

Bemerkungen:

Es ist zwar auch möglich, eine komplette Diskette nur mit den Befehlen **FORMAT** und **COPY** zu duplizieren, aber es ist eine sehr zeit- und vor allem auch nervenraubende Angelegenheit, weil jede Datei einzeln kopiert werden würde. DISKCOPY dagegen liest die Originaldiskette Track für Track und beschreibt auch die Kopie trackweise. Fehlerhafte Disketten oder kopiergeschützte Disketten können aber mit DISKCOPY nicht dupliziert werden.

Siehe auch:

BRU, COPY, FORMAT

PAG Sandini

DISKDOCTOR

Syntax:

2.0: DISKDOCTOR [DRIVE] <drive>:

2.1/3.0: Befehl nicht implementiert

Schablone:

2.0: DRIVE/A

2.1/3.0: Befehl nicht implementiert

Pfad:

C:

Funktion: PAG Sandini

Versucht bei fehlerhaften Disketten, möglichst viele Daten zu retten.

Beschreibung:

Daß auf Disketten von Zeit zu Zeit Fehler auftreten, läßt sich nicht vermeiden. Das ist auch ein Grund, warum der Befehl DISKCOPY (oben) etwas ausführlicher beschrieben wurde. Eine häufige Fehlerquelle ist ein Rechnerabsturz, Ausschalten oder Herausnehmen der Diskette, während eines Schreibvorganges. Der Befehl DISKDOCTOR kann aber unter Umständen viele Daten vor dem endgültigen Verlußt retten, aber nur, wenn er rechtzeitig eingesetzt wird, sobald eine Diskette anzeigt, daß etwas nicht mehr ganz korrekt ist (Übermäßig häufige Bewegungen des Schreib/Lese-Kopfes). Oder wenn der Amiga direkt darauf hinweist:

KEY <###> invalid, disk structure corrupt, use diskdoctor to correct it.

Die Syntax ist sehr einfach:

1> DISKDOCTOR df0:

Bevor Sie weiterlesen noch ein paar Hinweise:

- * DISKDOCTOR sollte immer das letzte Mittel sein - besonders für Amiga-Anwender, die nur ein Diskettenlaufwerk besitzen. Denn diese sind gezwungen, im einzigen verfügbaren Laufwerk mit der kaputten Diskette zu arbeiten, wenn dann etwas unvorhergesehenes passiert, kann das Arbeiten mit dem Rechner unmöglich werden. Es gibt - besonders für Amiga mit mehr als einem Laufwerk noch bessere Disketten-Retter aus dem Public-Domain-Bereich, beispielsweise "DISKSALV" von Dave Haynie.
- * Beim Arbeiten mit FFS-Disketten sollte DISKDOCTOR mit großer Vorsicht verwendet werden. Die meisten Anwender werden Ihre Festplatte im FFS (Fast File System) betreiben, so daß in der Regel hier die Probleme auftauchen, die dann noch gravierender sind. Es muß vor dem Aufruf von DISKDOCTOR sichergestellt sein, daß in der Datei "DEVS:Mountlist" der Eintrag "DOSTYPE" des betreffenden Gerätes den Wert 0x444F5301 (ASCII-Code für die Zeichen "DOS1") besitzt. Benutzen Sie nie DISKDOCTOR, wenn dieser Wert nicht überprüft wurde.

DISKDOCTOR kann bei seiner Arbeit sehr detaillierte Fehlermeldungen ausgeben, die im Folgenden in alphabetischer Reihenfolge erklärt werden. Sie sind nicht immer auf den ersten Blick verständlich, so daß eine Beschreibung mitunter häufiger zu lesen ist. Die Meldungen im Einzelnen:

"ATTENTION: Some file in directory <name> is unreadable und has been deleted."

Eine Datei des Unterverzeichnisses <name> wurde aus welchem Grund auch immer total zerstört, so daß DISKDOCTOR nicht einmal in der Lage war, den Namen der Datei zu erkennen. Solche Dateien können nicht mehr gerettet werden und werden von DISKDOCTOR automatisch entfernt, wobei obige Meldung erscheint.

"Block zero failed to format - Sorry!"

Unter äußerst ungünstigen Bedingungen kann nicht einmal die Spur 0 der Diskette formatiert werden, was ein Kennzeichen für eine total zerstörte Diskette ist. Ist dies der Fall, kann diese Diskette nicht mehr gerettet werden - auch nicht von anderen Programmen.

“Cannot write root block - Sorry!”

Ein weiterer gravierender Fehler tritt auf, wenn das Inhaltsverzeichnis der Diskette nicht mehr geschrieben werden kann. Aus diesem Verzeichnis heraus sollten eigentlich alle anderen Verzeichnisse und Dateien aufgefunden werden. Auch hier ist die Chance, die Diskette retten zu können, äußerst gering.

“Device <name> not found”

Das Gerät <name> konnte nicht gefunden werden, oder es existiert nicht. Die Ursache hierfür dürfte ein einfacher Tippfehler bei der Befehlseingabe von DISKDOCTOR sein.

“DiskDoctor cannot be run in background”

DISKDOCTOR ist ein interaktiver Befehl, ihn im Hintergrund laufen zu lassen (mit dem Befehl RUN), ist unmöglich. Wenn Sie unbedingt den Befehl als eigenen Prozess laufen lassen wollen, so müssen Sie eine eigene Shell starten.

PAG Sandini

“Disk must be write enabled”

DISKDOCTOR versucht, die komplette Diskette neu zu beschreiben - dies ist der tiefere Sinn seiner Philosophie. Vergewissern Sie sich also, daß die zu behandelnde Diskette nicht schreibgeschützt ist (das Loch in der Diskette muß durch den Schieber verschlossen sein).

“Disk type mismatch - formatting block zero”

Das ID-Kennzeichen (daran erkennt der Amiga AmigaDOS-Disketten) ist nicht vorhanden, oder kann nicht erkannt werden. In diesem Fall löscht DISKDOCTOR diese Spur komplett und fängt dann nochmal von vorne an. Dieser Fehler tritt oft bei Virus-Infektion auf, kann aber auch andere Ursachen haben.

“Delete corrupt files in dir <name>?”

DISKDOCTOR hat im Unterverzeichnis <name> einige fehlerhafte Dateien gefunden. Sie haben die Wahl, ob Sie diese unberührt lassen oder gleich entfernen wollen. Fehlerhafte Programme oder Befehle können gelöscht werden, da diese wohl nicht mehr laufen werden, während bei Text, Grafik und

Sound-Dateien selbst eine fehlerhafte Datei noch sehr hilfreich sein kann.

“Error: Unable to access disk”

Diese Meldung sagt nichts anderes als: “Sie haben vergessen, eine Diskette einzulegen, falls es Sie interessiert!”

“Failed to read key <###>”

Der Sektor <###> konnte nicht gelesen werden, möglicherweise auf Grund eines Hardware-Fehlers beim lesen.

“Failed to rewrite key <###>”

Der Sektor <###> konnte nicht wieder auf die Diskette zurückgeschrieben werden. Die Ursache hierfür ist meist ein physikalischer Fehler der Diskette (beispielsweise ein Kratzer).

“Hard error track <##>”

Die Spur <##> scheint mechanisch zerstört zu sein. Eine Behebung dieses Fehlers ist mit DISKDOCTOR unmöglich.

“Inserting dir <name>”

Das Verzeichnis <name> konnte wieder hergestellt werden und wird in das Hauptverzeichnis gesetzt.

“Inserting file <name>”

Die Datei <name> konnte wieder hergestellt werden und wird in das Hauptverzeichnis gesetzt. Dies geschieht, wenn das Verzeichnis, in dem die Datei ursprünglich gespeichert war, wegen eines Fehlers entfernt wurde.

“Key <###> of <name> out of range”

Die Adresse des Blocks <##> der Datei <name> liegt außerhalb des gültigen Bereiches der Diskette. Dieser Fehler kann bisweilen von DISKDOCTOR korrigiert werden, wenn der Befehl einige Bruchstücke der Datei zuordnen kann.

“Key <###> is unreadable”

Der Sektor kann nicht gelesen werden - möglicherweise ist er Teil eines me-

chanisch zerstörten Tracks.

“Not enough memory”

Dieser selten auftretende Fehler kann hingegen behoben werden, daß andere gerade laufende Programme beendet werden, damit wieder RAM-Speicher frei wird.

“Now copy files to a new disk an reformat this disk”

Mit diesem Satz verabschiedet sich DISKDOCTOR, nachdem er alles in seiner Macht stehende getan hat. Sie können nun nachsehen, ob Sie mit den geretteten Daten etwas anfangen können, und sollten die notwendigen Dateien gleich auf eine frische Diskette überspielen und die kranke neu formatieren. Treten beim Formatieren auch Fehler auf, ist diese Diskette schwer beschädigt und sollte schnellsten ausgesondert werden.

“Parent key of <#1> is <#2> which is invalid”

Der Sektor <#1> kann nicht in die Liste eingefügt werden, da der zuständige Hauptsektor <#2> irreparabel zerstört ist.

“Replacing dir <name>”

Das Verzeichnis <name> war nicht von einem Fehler betroffen und wird im alten Zustand wieder hergestellt.

“Root track failed to format - Sorry”

Der “Root track” (Spur 39, Oberseite der Diskette), auf dem alle notwendigen Informationen zu den Hauptverzeichnissen der Diskette stehen, konnte nicht formatiert werden. Hier hätte DISKDOCTOR versucht, einzelne Daten zu retten, obwohl das Inhaltsverzeichnis kaput ist, konnte aber nichts dagegen machen.

“Unable to read disk type formatting block zero”

Ist der Block 0, in dem wichtige Daten zur Diskette (Bootblock und Diskettentyp) stehen, fehlerhaft, versucht DISKDOCTOR den Block neu zu beschreiben und beginnt anschließend noch einmal von vorne.

“Unable to open disk.device”

Dies sollte eigentlich nie auftreten, da das “disk.device” als Teil des Betriebssystems normalerweise ständig im Speicher ist.

“Unable to write root - formatting root track”

Aufgrund eines Schreib-Lese-Fehlers der Spur des Inhaltsverzeichnisses versucht DISKDOCTOR das Inhaltsverzeichnis zu löschen und der Diskette den Namen “Lazarus” zu geben.

“Unexpected end of file”

Beim lesen einer Datei stellt sich heraus, daß diese kürzer ist, als sie den Angaben im “FileHeaderBlock” zufolge sein sollte. Ursachen könnten entweder Virus-Befall oder eine vorzeitige Herausnahme der Diskette während eines Schreibvorganges sein.

“Unknown device <name>”

Das Gerät <name> ist unbekannt, oder es existiert nicht. Die Ursache hierfür dürfte ein einfacher Tippfehler bei der Befehlseingabe von DISKDOCTOR sein.

“Warning: File <name> contains unreadable data”

Ein oder mehrere Blöcke der Datei <name> waren fehlerhaft. Die Dateien sind wahrscheinlich durch DISKDOCTOR verändert worden. Größte Vorsicht ist bei ausführbaren Programmen geboten, da diese mit hoher Wahrscheinlichkeit abstürzen.

“Warning: Loop detectet at file <name>”

Einige Adressen von Blöcken einer bestimmten Datei drehen sich im Kreis, d.h. der erste Block zeigt auf den zweiten, der auf einen dritten, welcher seinerseits wieder auf den ersten zeigt. Mögliche Ursache: Virus-Befall.

“<name> is not a device”

Bei der Befehlseingabe wurde DISKDOCTOR kein gültiger Gerätenamen übergeben.

Bemerkungen:

DISKDOCTOR kann auch benutzt werden, um eine versehentlich mit den Befehlen DELETE oder mit "FORMAT ... QUICK" gelöschte Dateien oder Disketten wieder herzustellen, sofern der Fehler rechtzeitig bemerkt wird und zwischen Löschvorgang und DISKDOCTOR kein schreibender Zugriff auf die Diskette erfolgte. Beide Befehle geben beim löschen lediglich den von den Daten der Datei (bzw. die gesamte Diskette bei FORMAT) belegten Speicher zum wiederbeschreiben frei, doch die Daten sind nachwievor auf der Diskette.

Es kann allerdings zu einem mittleren Chaos führen, wenn die Diskette vorher schon oftmals teilweise gelöscht und wieder beschrieben, anschließend wieder gelöscht.... wurde. DISKDOCTOR findet dann auch Teile älterer Dateien und holt auch diese wieder ans Tageslicht, wodurch der Inhalt einer Diskette sehr schnell undurchschaubar wird. Kopieren Sie vorher alle Dateien einzeln mit dem Befehl COPY von der Diskette, bevor sie sie zum DISKDOCTOR schicken.

Leider glaubten die Verantwortlichen von Commodore, daß DISKDOCTOR aufgrund seiner bisweilen sehr eigenwilligen Art, Daten zu retten, zu kostbarer Speicherplatz auf den Systemdisketten verbrauche und haben das Programm seit der Version 2.1 weggelassen. Mir selbst hat das Programm schon oft sehr viele Daten und Nerven gerettet. Vielleicht sollte sich Commodore Gedanken über einen gebührenden Nachfolger machen.

Siehe auch:

COPY, DELETE, DISKCOPY, FORMAT

DISPLAY

Syntax:

2.0: `DISPLAY [{<filename>|FROM <file>}] [OPT M|L|B|I|P|E|N|V]
[T <n>]`

2.1/3.0: Befehl nicht implementiert

Schablone:

2.0: `FILENAME/A/M, FROM/K, OPT/K, T/N`

2.1/3.0: Befehl nicht implementiert

Pfad:

Sys:Utilities

Funktion:

DISPLAY kann Bilder im IFF-ILBM-Format darstellen.

Beschreibung:

PAG Sandini

Mit dem Befehl DISPLAY kann sich der Anwender Bilder, die im IFF-ILBM-Format gespeichert sind anschauen. In der Befehlszeile können mehrere Dateien eingegeben werden, die dann in der angegebenen Reihenfolge gezeigt werden. Es ist auch möglich, alle Namen von Bilddateien zeilenweise in eine Datei zu schreiben, die dann in Verbindung mit dem Schlüsselwort FROM als "Programm" für die Bildreihenfolge dient.

Argumente:

FILENAME/A/M Hier werden die Namen der Dateien eingegeben, deren Bilder dargestellt werden sollen.

FROM/K Nach diesem Schlüsselwort folgt der Name einer Datei, in der zeilenweise die Namen der Bilddateien aufgeführt sind, die dann in dieser Reihenfolge gezeigt werden.

OPT/K Hier sind folgende Optionen möglich:

M (Mouseadvance) Wird diese Option gewählt, so wartet DISPLAY immer auf einen Tastendruck mit der Maus, bevor das nächste Bild

gezeigt wird. Die linke Maustaste ruft das nächste Bild in der Liste auf, die rechte Maustaste holt noch einmal das vorhergehende zurück. Diese Option kann mit der Option "T" (time) kombiniert werden.

- L (Loop) Hier wird das FROM-Programm solange wiederholt, bis es mit der Tastenkombination Ctrl-D abgebrochen wird. Kann auch mit "M" und "T" kombiniert werden.
- B (Backdrop) Hier wird DISPLAY gezwungen, den Bildschirm hinter allen anderen zu öffnen. Diese Option kann in Verbindung mit der Print-Option "P" sinnvoll verwendet werden.
- P (Print) Hier wird das angegebene Bild auf den in den Printer-Prefences-Programm eingestellten Drucker ausgegeben werden. Ausgelöst wird der Ausdruck selbst aber erst durch die Tastenkombination Ctrl-P, wenn das gewünschte Bild dargestellt wird.
- E (Extra half brite) Hier wird das Bild in den Bildschirmmodus EHB gezwungen.
- N (No transparent background) Schaltet eventuell vorhandenen durchsichtigen Hintergrund ab. Diese Option findet nur bei Arbeiten mit Video-Bildern Anwendung.
- V (oVerscan) Mit dieser Option können die Bilder in voller Bildschirmgröße angezeigt werden, bei denen zusätzlich der Bildschirmrand einbezogen wird.
- T/N (Time) Hier kann eingegeben werden, nach welcher Zeit (in Sekunden) ein neues Bild angezeigt werden soll. Bei dieser Option werden die Bilder nach der vorgesehenen Zeit automatisch gewechselt.

Bemerkung:

Ab der Workbench-Version 2.1 wurde dieses Programm durch das vielseitigere "Multiview" (siehe auch dort) ersetzt.

Siehe auch:

COLORS, MULTIVIEW

DPAT

Syntax:

2.0/2.1/3.0: DPAT [COM] <command> [PAT] <pattern> [DIR] <directory>
[<opt1>] [<opt2>] [<opt3>] [<opt4>]

Schablone:

2.0/2.1/3.0: COM/A,PAT/A,DIR/A,opt1,opt2,opt3,opt4

Pfad:

S:

Funktion:

Ermöglicht das sogenannte "Pattern matching" (die Verwendung von Jokerzeichen) auch für Programme, die von sich aus keine Jokerzeichen anerkennen.

Beschreibung: PAG Sandini

Dieser Befehl ist eigentlich gar keiner! Es handelt sich hier um ein sogenanntes Script, daß sich im Verzeichnis "S:" befindet. Hier wurden verschiedene AmigaDOS-Befehle zu einem Makro-Befehl zusammengefügt. Unter Verwendung des AmigaDOS-Befehls LIST können hier andere AmigaDOS-Befehle, die zwei Parameter (einen FROM- und einen TO-Parameter) benötigen und normalerweise keine Jokerzeichen akzeptieren (beispielsweise RENAME oder SORT), indirekt mit Jokerzeichen arbeiten.

DPAT rechnet mit drei übergebenen Argumenten, die anderen Optionen wurden noch eingebunden, falls ein über DPAT zu startender Befehl noch weitere Schalter oder Parameter benötigt.

Es sollte erwähnt werden, daß mit DPAT das aktuelle Verzeichnis nicht mit zwei Anführungszeichen "" und das Shell-Fenster nicht mit einem Stern * sondern mit zwei Sternen ** angesprochen werden kann.

Der Hintergrund dafür ist, daß DPAT seinerseits von dem Befehl EXECUTE ausgeführt wird, und dieser diese Zeichen anders interpretieren kann. Das aktuelle Verzeichnis kann ja über den Pfadnamen übergeben werden. Und wenn Sie die Ausgabe unbedingt in ein Konsolenfenster umleiten wollen, so sollten Sie dafür extra eines definieren "con:0/0...."

Argumente:

- COM/A Hier muß der Befehl eingegeben werden, der selbst keine Jokerzeichen kennt.
- PAT/A Hier wird das Namensmuster für die Dateien eingegeben, auf die der vorher eingetippte Befehl angewandt werden soll. Hier kann zwar auch ein Pfadname angegeben werden, doch die Jokerzeichen werden nur am Ende des Pfadnamens erwünscht, an der Stelle, an der normalerweise die konkreten Dateinamen stehen.
- DIR/A DPAT kann auch ein eigenes Zielverzeichnis verwenden, so daß die Dateien nicht nur mit dem angegebenen Befehl behandelt werden, sie kommen anschließend in dieses hier übergebene Verzeichnis. Es können nur Pfadnamen ohne Jokerzeichen verwendet werden.
- opt1...4 Für den Fall, daß der eigentliche Befehl weitere Argumente benötigt, können bis zu vier weitere Parameter übergeben werden. Diese Argumente werden dem eigentlich auszuführenden AmigaDOS-Befehl direkt übergeben, sind also auch an den Befehl fest gebunden. Sollten in sehr seltenen Fällen diese vier Parameter nicht ausreichen, können mehrere Argumente durch umfassende Anführungszeichen zu einer Option für DPAT zusammengefügt werden. Auf den richtigen Befehl hat dies keinen Einfluß.

Bemerkung:

Da aber die meisten AmigaDOS-Befehle mittlerweile auch die Jokerzeichen kennen, wird DPAT nur noch sehr selten benötigt. Weil es aber auf Diskette nicht viel Platz beansprucht und es ab und an durchaus sinnvolle Anwendungen gibt, sollte diese Datei verschont bleiben - möglicherweise auch deswegen, weil der Anwender hier einige interessante Anregungen zu den Script-Dateien erhält. Schauen Sie sich doch einmal die Datei an.

Siehe auch:

SPAT

ECHO

Syntax:

2.0/2.1/3.0: ECHO {string\$var} [NOLINE] [FIRST <n>] [LEN <n>] [TO <file>]

Schablone:

2.0/2.1/3.0: /M,NOLINE/S,FIRST/K/N,LEN/K/N,TO/K

Pfad:

in die Shell integriert

Funktion:

Gibt ein Zeichenkette ins Fenster oder eine Datei aus.

Beschreibung:

Die übergebene Zeichenkette wird durch den Befehl ECHO über das aktuelle Ausgabefenster oder -gerät ausgegeben. Voreingestellt ist hier die Ausgabe in das Shell-Fenster, unter dessen Prozess dieser Befehl aufgerufen wurde, es kann aber auch in jedes andere Gerät oder eine Datei durch eine Ausgabeumleitung geschrieben werden. Enthält die auszugebende Zeichenkette Leerzeichen, so sollte die gesamte Zeichenkette in Anführungszeichen gesetzt werden. Ohne irgenwelche zusätzlichen Argumente erzeugt ECHO eine Leerzeile.

Argumente:

string\$var/M Zeichenkette oder Variable einer Script-Datei, die ausgegeben werden soll. Es gibt neben den normalen druckbaren Zeichen noch verschiedene andere Möglichkeiten, auf die Ausgabe Einfluß zu nehmen, wie unten angeführte Beispiele zeigen.

NOLINE/S Normalerweise setzt ECHO nach der Ausgabe der Zeichenkette ein RETURN-Zeichen, so daß die Ausgabe in der nächsten Zeile fortgesetzt wird. Wenn jedoch dieses Schlüsselwort verwendet wird, so bleibt der Cursor nach der Zeichenkette stehen.

- FIRST/K/N** Diese Zahl gibt an, ab welchen Zeichen der übergebenen Zeichenkette - oder Variablen - die Ausgabe beginnen soll. Dabei wird das erste Zeichen als Zeichen Null gezählt. Wird hier die Zahl 256 übergeben, so druckt ECHO das letzte Zeichen der Zeichenkette.
- LEN/K/N** Mit der Zahl LEN kann angegeben werden, wieviele Zeichen - von rechts aus gezählt - ausgegeben werden sollen, falls LEN ohne FIRST verwendet wird. ECHO gibt dann die letzten <n> Zeichen der Zeichenkette aus. Wird LEN jedoch in Verbindung mit FIRST verwendet, so werden LEN=<n> Zeichen beginnend beim Zeichen FIRST=<n> ausgegeben.
- TO/K** Ermöglicht eine Umleitung der Ausgabe in eine beliebige Datei oder ein Gerät, dessen Name hier eingegeben werden muß.

Beispiele:

PAG Sandini

Um nur die letzten Zeichen einer Zeichenkette zu erhalten, wird der Parameter LEN alleine eingesetzt:

```
1.Work:DOS3.0> echo abcdefghijklmn len=4
klmn
```

Es gibt verschiedene Möglichkeiten, auf die Ausgabe Einfluß zu nehmen. In einer Zeichenkette kann mit **"*N"** ein Zeilenvorschub erzwungen werden, und wird vor normalerweise nicht druckbaren Zeichen ein Stern gesetzt, so wird das folgende Zeichen unverändert ausgegeben: 1.Work:DOS3.0> ECHO **"**"Dies ist ein erster Test ***

"Dies ist ein erster Test "

```
1.Work:DOS3.0> ECHO Eins Zwei Drei
Eins Zwei Drei
```

```
1.Work:DOS3.0> ECHO "Auch der Stern kann angezeigt werden: **"
Auch der Stern kann angezeigt werden: *
```

Mit nur einem einzigen Befehl kann ein ganzer Absatz geschrieben werden:

1.Work:DOS3.0> ECHO “*N*N*NDie letzten Worte eines Autors:*N*N*
*”Du kannst ruhig ausschalten, ich habe den Text schon gespeichert!
*”*N*N*NAus: Murphy’s Gesetze”

Die letzten Worte eines Autors:

“Du kannst ruhig ausschalten, ich habe den Text schon gespeichert!”

Aus: Murphy’s Gesetze

Bemerkung:

ECHO ist für die Verwendung in Script-Dateien vorgesehen. Es können bei ECHO auch Escape-Sequenzen (siehe obige Tabelle) verwendet werden. Nur muß in diesem Fall das Esc-Zeichen bzw. die entsprechende Taste durch die Zeichenfolge “*E” ersetzt werden.

Es ist zwar möglich, die Ausgabe von ECHO in eine Datei, die anzugeben ist, umzuleiten, der umgekehrte Weg funktioniert aber nicht. Es kann nur unter sehr großen Bemühungen eine Zeichenkette aus einer Datei eingegeben werden, wenn gleichzeitig mit dem Fragezeichen “?” am Ende der Befehlszeile die Hilfsfunktion abgerufen wird. Hierbei kommt aber (zumindest bei meinen Versuchen) der Befehl ECHO gehörig ins Schleudern, wenn gleichzeitig noch die Parameter FIRST und LEN verwendet werden.

Siehe auch:

TYPE

ED

Syntax:

2.0/2.1/3.0: ED [FROM] <filename> [[SIZE] <nn>] [WITH <macrofile>]
 [WINDOW <window>] [[TABS] <n>] [[WIDTHCOLS] <n>]
 [[HEIGHTROWS] <n>]

Schablone:

2.0/2.1/3.0: FROM/A,SIZE/N,WITH/K,WINDOW/K,TABS/N,WIDTH=COLS/N,
 HEIGHT=ROWS/N

Pfad:

C:

Funktion:

Startet Texteditor "ED"

Beschreibung:

ED ist einer der drei beim Amiga mitgelieferten Texteditoren, der bei sämtlichen Versionen verfügbar ist. ED bietet einige sehr einfache Manipulationsmöglichkeiten für reine ASCII-Texte. Ein weiterer wesentlich umfangreicherer Editor ist MicroEMACS, der später beschrieben wird.

ED unterstützt die Funktionstasten je mit und ohne Shift-Taste, so daß hier 20 Befehle vom Anwender selbst belegt werden können. Die Voreingestellte Funktionstastenbelegung befindet sich in der Datei "S:ED-Startup". ED verfügt auch über eine ARexx-Schnittstelle namens "ED".

Argumente:

FROM/A ED muß wissen, welche Datei Sie bearbeiten wollen, also muß der entsprechende Dateiname eingegeben werden. Auch wenn die Datei noch nicht existiert, muß der künftige Name hier eingetippt werden.

SIZE/N Hier kann die maximale Dateigröße angegeben werden, mit der ED arbeiten soll. Diese Zahl benutzt ED bei der

Anforderung des Textspeichers vom System. Diese Zahl sollte groß genug gewählt werden, damit ED vernünftig arbeiten kann. Wird keine Zahl genannt, so gilt der voreingestellte Wert von 40000 Bytes. Dieser Wert sollte nicht unterschritten werden.

WITH/K

Beim Start versucht ED sein Menü und die Funktionstastenbelegung aus der Datei "S:ED-Startup" zu laden. Wenn Sie jedoch eine eigene Konfigurations-Datei erstellt haben, können Sie diese hier angeben. In dieser Datei können auch Makro-Befehle definiert werden.

WINDOW/K

Hiermit kann das Fenster von ED nach eigenem Ermessen gestaltet werden - natürlich nur innerhalb der Systembedingten Grenzen. Sie können den Editor so beispielsweise auch im Shell-Fenster starten, oder auch auf der seriellen Schnittstelle AUX:, so daß ein anderer Computer ED steuern kann.

WIDTH/K

Hier wird die Breite des Fensters in Anzahl von Zeichen angegeben. Die Zahl hat keinen Einfluß auf die tatsächliche Breite des Fensters, sondern bezieht sich auf die Breite des Textes. Das Schlüsselwort WIDTH muß angegeben werden, wenn hier eine Zahlenangabe gewünscht wird.

HEIGHT/K

Analog zu WIDTH wird hier die Höhe des Fensters in Anzahl der Zeilen angegeben. Auch HEIGHT ist ein notwendiges Schlüsselwort.

TABS/N

Diese Zahl definiert die Anzahl von Leerzeichen, die beim Drücken der Tabulatortaste eingefügt werden. Voreingestellt ist der Wert 3. ED kennt keine richtigen Tabulatorzeichen.

Siehe auch:

EDIT, MicroEMACS

EDIT

Syntax:

2.0/2.1/3.0: EDIT [FROM] <filename> [[TO] <filename>] [WITH <macrofile>]
[VER <window>] [[OPT P <lines>|W <chars>] | PREVIOUS <lines>|
WIDTH <chars>]

Schablone:

2.0/2.1/3.0: FROM/A, TO, WITH/K, VER/K, OPT/K, WIDTH/N, PREVIOUS/N

Pfad:

C:

Funktion:

Startet den Zeileneditor EDIT.

Beschreibung:

EDIT ist ein sehr gewöhnungsbedürftiger zeilenorientierter Texteditor mit sehr ähnlichen Eigenschaften wie ED und vergleichbar zu den Zeileneditoren der MS-DOS-Rechner. EDIT besitzt kein Fenster, geschweige denn ein Menü. Sämtliche Eingaben - auch die der Editor-Befehle - müssen über die Tastatur erfolgen. Dies hört sich sehr umständlich und veraltet an, kann aber bisweilen überraschend hilfreich sein.

Aufgrund der Tatsache, daß EDIT die zu bearbeitende Textdatei nicht komplett lädt, können auch umfangreichste Texte bearbeitet werden, bei denen andere Texteditoren (evtl. schon aufgrund des eigenen Speicherbedarfs) schon längst kapituliert haben. Ein weiterer Vorteil ist, daß Binärdaten, die in der einen oder anderen Datei vorhanden sind, durch EDIT nicht verändert werden, weil diese Zeilen nicht gelesen bzw. angezeigt werden müssen.

Alle Befehle und Funktionen von EDIT werden im folgenden erklärt.

Argumente:

FROM/A Dieser Parameter ist der einzige, auf den EDIT unbedingt besteht. EDIT braucht den Namen einer schon existierenden Datei, die zu bearbeiten ist. Sollte noch keine Datei

exisitieren, so suchen Sie sich eine beliebige kurze schon vorhandene Datei aus, und arbeiten diese um. Die so umgearbeitete Datei kann von EDIT unter einem neuen Namen abgespeichert werden - siehe Argument "TO".

TO

Dieses optional anzugebende Schlüsselwort wird benutzt, um EDIT mitzuteilen, daß die Quelldatei unverändert bleiben soll und sämtliche Änderungen in der hier angegebenen Datei abgespeichert werden sollen.

WITH/K

Hier kann der Name einer Datei mit EDIT-Befehlen angegeben werden die unmittelbar nach dem Start von EDIT auszuführen sind. Dieses Argument kann dazu verwendet werden, EDIT zu automatisieren, so daß die Änderung einer Datei nach vorgegebenen Mustern durchgeführt werden kann. In diesem Fall sollte die Datei mit dem "W"-Befehl des EDIT enden.

VER/K

Dieses Schlüsselwort sagt EDIT, es solle die Meldungen nicht wie sonst üblich ins Shell-Fenster sondern in das hier genannte Gerät oder eine Datei schreiben.

OPT/K

Die beiden möglichen Optionen haben die gleiche Funktion, wie die folgenden Argumente PREVIOUS und WIDTH. Sie sollten jedoch unter AmigaDOS 2.0/2.1 nicht mit den ausgeschriebenen Schlüsselwörtern gemischt eingesetzt werden.

PREVIOUS/K

Über dieses Schlüsselwort geben Sie die Anzahl der möglichen zwischengespeicherten vorhergehenden Zeilen an. Voreingestellt ist der Wert 40.

WIDTH/K

Durch dieses Schlüsselwort wird die maximale Zeilenlänge festgelegt. EDIT multipliziert die Werte PREVIOUS und WIDTH, um die Größe des erforderlichen Textspeichers zu ermitteln, welcher dann vom Betriebssystem angefordert wird. Hier wurde der Wert 120 voreingestellt.

EDIT-Befehle:

Hier können wir nur eine knappe aber ausreichende Beschreibung aller EDIT-Befehle geben. Um die einzelnen Befehle verstehen zu können, sollten Sie damit herumexperimentieren - aber bitte nur mit Dateien, die keine wertvollen Daten enthält. Manche Amiga-Profis sind überzeugt, daß ein Anwender, der mit EDIT umgehen kann, schneller und besser arbeiten kann als mit ED. Sollte kein Verzeichnis ":T" oder das Gerät "T:" existieren, so schafft EDIT eines.

Befehl	Bedeutung
'	Wiederholt den vorhergegangenen Befehl. Nicht nach A, B oder E anwenden.
=n	Gehe zu Zeile n, setze dort Bearbeitung fort.
:	Trennt Befehle, wenn in einer Zeile mehr als nur ein Befehl eingegeben werden soll (analog zum Doppelpunkt in BASIC oder Semikolon ; in C)
n(...)	Der in Klammern eingeschlossene Befehl wird n-mal wiederholt. Für n muß eine Zahl eingegeben werden. Wird 0 eingegeben, so wird der Befehl unendlich oft wiederholt oder abgebrochen, wenn ein Fehler auftritt.
n<	Setzt den Cursor in der aktuellen Zeile um n Zeichen nach links. Wird keine Zahl eingegeben, so wird der Cursor um ein Zeichen versetzt.
n>	Setzt den Cursor in der aktuellen Zeile um n Zeichen nach rechts. Wird keine Zahl eingegeben, so wird der Cursor um ein Zeichen versetzt.
?	Zeigt die aktuelle Zeile in einfacher Darstellung an.
!	Zeigt die aktuelle Zeile an, auch nicht druckbare Zeichen und unterstreicht Großbuchstaben.
n#	Löscht n Zeichen ab der aktuellen Cursor-Position. Wurde keine Zahl eingegeben, wird nur ein Zeichen gelöscht.

- n\$** Macht n Zeichen ab der aktuellen Cursor-Position klein. Wurde keine Zahl eingegeben, wird nur ein Zeichen geändert.
- n%** Macht n Zeichen ab der aktuellen Cursor-Position groß. Wurde keine Zahl eingegeben, wird nur ein Zeichen geändert.
- n_** Überschreibt ab der aktuellen Cursorposition n Zeichen mit Leerzeichen. Wurde keine Zahl eingegeben, wird nur ein Zeichen überschrieben.
- A/string/text** Sucht in der aktuellen Zeile nach der Zeichenkette string und fügt anschließend text ein. Die Position des Cursor wird dabei nicht verändert.
- AP/string/text** Sucht in der aktuellen Zeile nach der Zeichenkette string und fügt anschließend text ein. Die Position des Cursor wird dabei an die Stelle gesetzt, an der der Text eingefügt wurde.
- B/string/text** Sucht rückwärts in der aktuellen Zeile nach der Zeichenkette string und fügt vorher text ein. Die Position des Cursor wird dabei nicht verändert.
- BP/string/text** Sucht rückwärts in der aktuellen Zeile nach der Zeichenkette string und fügt vorher text ein. Die Position des Cursor wird dabei an die Stelle gesetzt, an der der Text eingefügt wurde.
- BF/string/** Sucht in der Zeile rückwärts nach der Zeichenkette string. Wird keine gefunden, so werden alle vorherigen Zeilen rückwärts durchsucht, bis entweder eine passende Zeichenkette gefunden wurde, oder der Dateianfang erreicht wurde.
- C.text.** Führt die Befehlsdatei "text" aus. Bemerkung: die beiden Punkte schließen immer einen Dateinamen ein.
- CF.text.** Schließt eine geöffnete Datei namens "text".
- CGn** Löscht die globale Operation Nr. n aus dem Arbeitsspeicher, aber macht diese nicht rückgängig. Wenn keine Zahl angegeben wird, werden alle Operationen gelöscht. Siehe auch SHG.

CL/string/	Verbindet die aktuelle Zeile mit der folgenden Zeile und setzt die Zeichenkette string dazwischen. wird nur "CL/" verwendet, werden beide Zeilen direkt miteinander verbunden.
D	Löscht die aktuelle Zeile. (Nicht mit dem Befehl # verwechseln, der einzelne Zeichen löscht. Siehe auch DF, #)
DF/string/	Sucht ausgehend von der aktuellen Cursorposition die Zeichenkette string und löscht die Zeile, sobald er gefunden wurde. Wird in der Zeile kein string gefunden, so wird die Datei durchsucht, bis entweder die Zeichenkette auftaucht oder das Dateiende erreicht wird. Vorsichtig anwenden! (Siehe auch DFA)
DFA/string/	Löscht in der aktuellen Zeile alles nach der Zeichenkette string. Ist die Zeichenkette nicht vorhanden, wird eine Fehlermeldung ausgegeben.
DFB/string/	Löscht in der aktuellen Zeile alles ab der Zeichenkette string. Ist die Zeichenkette nicht vorhanden, wird eine Fehlermeldung ausgegeben.
DGn	Schaltet den globalen Befehl Nr. n vorübergehend aus.
DTA/string/	Löscht vom Beginn der Zeile bis zum Ende der Zeichenkette string.
DTB/string/	Löscht vom Beginn der Zeile bis zum Beginn der Zeichenkette string.
E/string/text	Sucht string und ersetzt es mit text.
EGn	Schaltet den globalen Befehl Nr. n wieder ein.
EP/string/text	Sucht string und ersetzt es mit text. Der Cursor wird an die geänderte Stelle gesetzt.
F/string/	Sucht die Zeichenkette string und setzt dort den Cursor hin. Wird EDIT nicht in der Zeile fündig, durchsucht er die ganze

Datei ab dieser Zeile bis die Zeichenkette auftaucht oder das Dateiende erreicht wird.

FROM	Fügt an der aktuellen Cursorposition eine Datei ein.
FROM.text.	Fügt an der aktuellen Cursorposition die Datei "text" ein.
TO	Schickt die Ausgabe zu der in der Befehlszeile beim Start von EDIT angegebenen Datei oder das genannte Gerät.
TO.text.	Schickt die Ausgabe in die Datei "text".
GA/string/text	Fügt in der ganzen Datei nach der Zeichenkette string die Zeichenkette text ein.
GB/string/text	Fügt in der ganzen Datei vor der Zeichenkette string die Zeichenkette text ein.
GE/string/text	Ersetzt in der ganzen Datei die Zeichenkette string durch die Zeichenkette text.
Hn	Setzt an der Zeile n eine Halte-Marke. Diese gilt, bis die Zeile erreicht wird.
I	Füge Zeichen von der Shell ein. Zum Abbruch muß Ctrl-C und Return gedrückt werden.
I.text.	Fügt Text aus der Datei "text" ein.
Mn	Geht um n Zeilen weiter.
M+	Geht zu der untersten Zeile, die im Speicher ist (muß nicht das Ende der Datei sein).
M-	Geht zu der obersten Zeile, die im Speicher ist (muß nicht der Beginn der Datei sein).
N	Geht zur nächsten Zeile.

P	Geht zur vorhergehenden Zeile.
PA/string/	Setzt den Cursor im Anschluß an die Zeichenkette string.
PB/string/	Setzt den Cursor vor die Zeichenkette string.
PR	Setzt den Cursor an den Zeilenanfang.
Q	Beendet EDIT ohne abzuspeichern.
R	Holt Zeichen von der Shell und überschreibt damit die schon vorhandenen Zeichen. Zum Abbruch muß Ctrl-C und Return gedrückt werden. R.text. Holt Zeichen von der Datei text und überschreibt damit die schon vorhandenen Zeichen. Zum Abbruch muß Ctrl-C und Return gedrückt werden.
REWIND	Setzt Cursor an den Anfang der Datei.
SA/string/	Teilt die Zeile nach der Zeichenkette string. Der Cursor befindet sich am Anfang der neuen Zeile.
SB/string/	Teilt die Zeile vor der Zeichenkette string. Der Cursor befindet sich am Anfang der neuen Zeile.
SHD	Zeigt die Liste der eingestellten Daten.
SHG	Zeigt die Liste der eingestellten globalen Befehle.
STOP	Beendet EDIT und läßt die Originaldatei unverändert.
T	Zeigt die Datei ab der aktuellen Zeile bis zum Dateiende an.
Tn	Zeigt n Zeilen ab der aktuellen Zeile an.
TLn	Wie T, hier werden jedoch zusätzlich Zeilennummern angezeigt.
TN	Zeigt die Zeilen bis zum Pufferende.

TP	Zeigt die vorhergehenden Zeilen bis zum Pufferbeginn.
TR+	Führende Leerzeichen werden entfernt.
TR-	Führende Leerzeichen werden nicht entfernt.
V+	Die Zeile wird nach jeder Befehlsausführung angezeigt.
V-	Die Zeile wird nicht nach jeder Befehlsausführung angezeigt.
W	Speichert die Änderungen und verläßt EDIT.
Z text	Ändert das Abschlußzeichen auf "text".

Siehe auch:

ED, MicroEMACS

PAG Sandini

ELSE

Syntax:

2.0/2.1/3.0: ELSE

Schablone:

2.0/2.1/3.0: -

Pfad:

in die Shell integriert

Funktion:

In einer If-Anweisung werden die alternativen Befehle ausgeführt.

Beschreibung:

ELSE kann nur in Scriptdateien verwendet werden. Er ist ein Teil der Anweisungen IF, ELSE, ENDIF. Es wird mit IF eine bestimmte Bedingung geprüft, trifft diese nicht zu, werden die Befehle ausgeführt, die nach ELSE und vor ENDIF stehen. Der Befehl hat keine Argumente und ist außerhalb einer IF-ENDIF-Anweisung bedeutungslos. In geschachtelten IF-ELSE-ENDIF-Anweisungen bezieht sich ELSE immer auf die zuletzt ausgeführte IF-Abfrage.

Beispiel:

Auszug einer möglichen Script-Datei:

```
IF EXISTS RAM:Env  
ECHO "Ram:Env existiert bereits"  
ELSE  
ECHO "Ram:ENV nicht gefunden!"  
ENDIF
```

Siehe auch:

IF, ENDIF

ENDCLI

Syntax:

2.0/2.1/3.0: ENDCLI

Schablone:

2.0/2.1/3.0: -

Pfad:

in die Shell integriert

Funktion:

Schließt das Shell-Fenster und beendet den Prozess.

Beschreibung:

Dieser Befehl schließt das Shell-Fenster und beendet den Prozess. Der Befehl ENDSHELL ist gleichwertig. Wird die kurz zu sehende Meldung "CLI ## ending" nicht gewünscht, kann die Ausgabe ins Nichts umgeleitet werden:

1> ENDCLI >NIL:

Siehe auch:

ENDSHELL, NEWCLI

ENDIF

Syntax:

2.0/2.1/3.0: ENDIF

Schablone:

2.0/2.1/3.0: -

Pfad:

in die Shell integriert

Funktion:

Beendet den IF-Block in einer Script-Datei

Beschreibung:

ENDIF ist der Abschlußbefehl für den IF-ENDIF- oder IF-ELSE-ENDIF-Block und schließt die zuletzt ausgeführte IF-Anweisung ab. Er kann nur in Script-Dateien verwendet werden und ist außerhalb völlig bedeutungslos.

Siehe auch:

IF, ELSE

ENDSHELL

Syntax:

2.0/2.1/3.0: ENDSHELL

Schablone:

2.0/2.1/3.0: -

Pfad:

in die Shell integriert

Funktion:

Schließt das Shell-Fenster und beendet den Prozess.

Beschreibung:

Dieser Befehl schließt das Shell-Fenster und beendet den Prozess. Der Befehl ENDCLI ist gleichwertig. Wird die kurz zu sehende Meldung "CLI ## ending" nicht gewünscht, kann die Ausgabe ins Nichts umgeleitet werden:

1> ENSHELL >NIL:

Bemerkungen:

ENDSHELL kann immer ausgeführt werden, es kann nur bisweilen vorkommen, daß ein Shell-Fenster noch nicht geschlossen werden kann, weil ein anderer Befehl noch mit RUN gestartet wurde.

Siehe auch:

ENDCLI, NEWCLI

ENDSKIP

Syntax:

2.0/2.1/3.0: ENDSKIP

Schablone:

2.0/2.1/3.0: -

Pfad:

in die Shell integriert

Funktion:

Der SKIP-Block wird in einer Befehlsdatei beendet.

Beschreibung:

Mit dem SKIP-Block können bei der Abarbeitung von Script-Dateien Befehle übersprungen werden. Wird dieser Befehl bei der Bearbeitung der Datei angetroffen, so werden die nachfolgenden Befehle wieder ausgeführt. Gleichzeitig wird eine Warnung (WARN, Code 5) erzeugt, womit eine Fehlerbehandlung bei nicht gefundenen Sprungmarken ermöglicht wird.

Siehe auch:

SKIP, LAB

EVAL

Syntax:

2.0/2.1/3.0: EVAL [VALUE1] <value1> {[OP] <operator> [[VALUE2] <value2>]}
[TO <file>] [LFORMAT <format>]

Schablone: 2.0/2.1/3.0: VALUE1/A,OP,VALUE2/M,TO/K,LFORMAT/K

Pfad:

C:

Funktion:

Wertet einfache Ausdrücke aus.

Beschreibung:

EVAL ist einer der AmigaDOS-Befehle, die am wenigsten durchschaut werden, dabei ist gerade EVAL in Script-Dateien sehr hilfreich, da er neben sehr einfachen Ganzzahl-Rechenoperationen auch Zahlenbasen umwandeln kann. Es ist aber nicht ganz so einfach, Zahlen anderer Basen als das Dezimalsystem, einzugeben:

Eine Hexadezimalzahl wird durch ein führendes "0x" oder "#x" gekennzeichnet, eine Oktalzahl durch "0" oder "#0" vorne weg. Es ist auch möglich, den ASCII-Code eines Zeichens zu verwenden, wenn vor dem Zeichen ein Apostroph steht. Dabei muß darauf geachtet werden, daß eine Dezimalzahl nicht mit einer führenden Null beginnen darf, da sie sonst falsch interpretiert wird. Die Anwendung ist relativ einfach, wie folgende Beispiele zeigen:

```
5> EVAL 0xFF
255
5> EVAL 'c
99
5> EVAL 077
63
5> EVAL 2+2
4
```

Es können sovieler numerische Argumente übergeben werden, wie in eine Befehls-Zeile passen, Leerzeichen zwischen den Zahlen und den Operanden sind nicht notwendig. Der Befehl EVAL behandelt die Rechnung nach den mathematischen Regeln der Priorität von Klammern vor Multiplikation/Division vor Addition/Subtraktion. EVAL kann auch Environment-Variablen durch voranstellen des Dollar-Zeichens verwenden:

5> EVAL \$Zahl + \$Zahl 16

Seit der Version 2.0 ist es auch möglich, den Wert einer Environment-Variable bei der Rechnung zu verwenden und das Ergebnis wieder dieser Variablen zuzuordnen:

5> EVAL \$Zahl + 1 to ENV:Zahl

oder komplexer (aber erlaubt):

5> EVAL (\$Zahl + \$Wert) * 2 to ENV:Zahl

Argumente:

VALUE1/A Der einzige Parameter, den EVAL unbedingt benötigt. Es können alle oben genannten Wertarten eingetippt werden.

OP Die Operation, die auf den (oder die) eingegebenen Wert(e) angewendet wird. Erlaubt sind:

- +** Addition
- Subtraktion
- *** Multiplikation
- /** Division
- MOD** Modulo-Arithmetik
- &** logisch Und
- |** logisch Oder
- ~** logisch Nicht
- LSH** bitweises Schieben nach links
- RSH** bitweises Schieben nach rechts
- Negation
- XOR** exklusives Oder
- EQV** bitweise Equivalenz
- ()** Klammern zum Gruppieren von Termen.

VALUE2/M	Der zweite Wert, der EVAL übergeben wird.
TO/K	Name einer Datei, in die das Ergebnis geschrieben werden soll. Dieses Argument ist sehr wichtig, wenn EVAL in Scriptdateien verwendet werden soll, da eine Umleitung in eine Datei der Zuweisung eines Wertes an eine Variable entspricht. Über eine normale Ausgabeumleitung mit ">Ziel" ist es nicht möglich, gleichzeitig eine Variable zu lesen und wieder zu beschreiben. (In diesem Fall versucht AmigaDOS diese eine Datei gleichzeitig einmal zum Lesen und einmal zum Schreiben zu öffnen, was natürlich beim zweiten Öffnungsversuch scheitert.)
LFORMAT/K	<p>Hier kann die Ausgabe von EVAL stark beeinflußt werden, ähnlich der Ausgabe des LIST-Befehles (siehe auch dort). Für C-Programmierer dürfte dies keine großartige Neuigkeit sein. Sie können eine ganze Zeichenkette eingeben, EVAL gibt dann die Antwort als vollständigen Satz. An der Stelle, an der das Ergebnis in der Zeichenkette stehen soll, wird durch folgende Zeichen gekennzeichnet, die gleichzeitig die Basis der Zahl angeben (wie bei der Programmiersprache C):</p> <ul style="list-style-type: none">%x Das Resultat wird als Hexadezimalzahl ausgegeben.%n Das Resultat wird als Dezimalzahl ausgegeben.%o Das Resultat wird als Oktalzahl ausgegeben.%c Das Resultat wird als ASCII-Zeichen ausgegeben. <p>Werden große Buchstaben verwendet, können zusätzlich noch die Anzahl der Stellen bestimmt werden: "%X4" gibt die Zahl als vierstellige Hexadezimalzahl aus.</p>

Bemerkung:

EVAL kann Zahlen bis zu einer Größe von 32 Bit verarbeiten, also einen Zahlenbereich von -2147483646 bis +2147483646.

Siehe auch:

LIST

EXCHANGE

Syntax:

2.0/2.1/3.0: EXCHANGE [CX_PRIORITY <priority>] [CX_POPUP <yes/no>]
 [CX_POPKEY <hotkeys>]

Schablone:

2.0/2.1/3.0: CX_PRIORITY/K/N,CX_POPUP/K,CX_POPKEY/K

Pfad:

2.0: Sys:Utilities
2.1/3.0: Extras:Tools/Commodities

Funktion:

Steuert und überwacht die Commodities-Programme, die im Hintergrund laufen.

PAG Sandini

Beschreibung:

EXCHANGE startet das gleichnamige Programm zur Steuerung und Überwachung von Commodities. EXCHANGE erlaubt Ihnen, im Hintergrund laufende Commodities zu ändern oder gar aus dem Speicher zu entfernen. Normalerweise sind zusätzliche Argumente bei der Befehlseingabe unnötig.

Argumente:

Siehe Beschreibung zum Befehl CLICKTOFRONT.

Bemerkungen:

Wird bei irgendeinem Commodity-Programm (EXCHANGE oder eines der unten genannten) eine Befehlszeile nicht korrekt eingetippt, so geben die Programme keine Fehlermeldung aus. Sie können aber mit dem Programm STATUS (siehe auch dort) testen, ob das betreffende Commodity richtig gestartet werden konnte.

Siehe auch:

AUTOPOINT, BLANKER, CLICKTOFRONT, FKEY, IHELP, NOCAPSLOCK

EXECUTE

Syntax:

2.0/2.1/3.0: EXECUTE <Script-Datei> [{argumente}]

Schablone:

2.0/2.1/3.0: hängt nur von der Script-Datei ab.

Pfad:

C:

Funktion:

Führt eine Script-Datei aus.

Beschreibung:

Dieser Befehl wird dazu benutzt, sogenannte Scripts (Befehlsdateien) auszuführen, die AmigaDOS-Befehle enthalten. Die einzelnen Zeilen dieser Datei werden wie Befehlszeilen ausgeführt, die in der Shell hintereinander nach der Eingabeaufforderung eingegeben worden wären. Hat eine Befehlsdatei das Schutzbit "S" (siehe dazu auch den Befehl PROTECT) gesetzt, und befindet sich diese im aktuellen Suchpfad, so braucht der Befehl EXECUTE nicht eingegeben werden. Diese Befehlsdatei läßt sich dann so starten, wie ein normaler AmigaDOS-Befehl.

Es ist möglich, durch Verwendung bestimmter Schlüsselwörter der Befehlsdatei auch Argumente zu übergeben, die dann einfach hinter dem Namen der Befehlsdatei beim Befehl EXECUTE eingetippt werden. Beim Aufruf des Scripts werden die angegebenen Argumente vor dem Start noch mit den von der Befehlsdatei geforderten Liste von Argumenten verglichen, stimmen Anforderung und tatsächlichen Argumente in ihrer Art überein, werden die Werte in die Variablen des Scripts übernommen und die Befehlsbearbeitung beginnt. Es ist auch möglich, Voreinstellungen vorzunehmen, für den Fall, daß der Anwender keine Parameter angibt. Gibt es weder einen Parameter vom Anwender noch einen Vorgabewert, so bleibt der Inhalt der entsprechenden Variablen leer, jedesmal wenn sie im Script auftaucht, wird sie ignoriert (besser: ersatzlos gestrichen). Ist der Wert einer Variablen bekannt, so wird

diese bei jedem Auftreten in der Befehlsdatei durch den Wert ersetzt, bevor ein AmigaDOS-Befehl ausgeführt wird.

Die Schlüsselwörter für die Parameterübergabe und die Möglichkeiten, wie Variablen verwendet werden können, werden im Folgenden aufgezeigt:

- < und > Diese spitzen Klammer werden verwendet, um die Variablennamen zu kennzeichnen. Jedes Wort in solchen Klammern wird von Execute als Variable interpretiert und vor der Ausführung eines Befehls durch den Wert der Variablen ersetzt. Da die spitzen Klammern aber bisweilen andere Bedeutungen übernehmen sollen, können diese mit den Schlüsselwörtern ".BRA" und ".KET" durch andere Zeichen ersetzt werden.
- .BRA <char> Ersetzt die öffnende spitze Klammer für die Kennzeichnung des Beginns eines Variablennamens durch das angegebene Zeichen <char>. Empfohlen wird hier die geschweifte Klammer "{", da diese bislang nirgends verwendet wird.
- .KET <char> Ersetzt die schließende spitze Klammer für die Kennzeichnung des Endes eines Variablennamens durch das angegebene Zeichen <char>. Empfohlen wird auch hier die geschweifte Klammer "}".
- .DOL <char> Hierdurch wird das Dollarzeichen "\$" durch ein anderes anzugebendes Zeichen <char> ersetzt. Das Dollarzeichen wird dazu verwendet, hinter Variablennamen den Voreinstellungswert festzulegen. Die Beispiele später bringen wohl eher Licht ins Dunkel. Für ".DOL" kann auch ".DOLLAR" geschrieben werden.
- .DOT <char> Hierdurch wird der Punkt, der eigentlich die anderen Schlüsselwörter einleitet, durch das anzugebende Zeichen <char> ersetzt. Alle nachfolgenden Schlüsselwörter werden mit diesem Zeichen eingeleitet, sonst werden sie nicht von EXECUTE als solche anerkannt.

.KEY <args>

Dies ist das wichtigste Schlüsselwort, denn es legt fest, welche Argumente die Befehlsdatei vom Anwender erwartet. Es gibt sozusagen die Syntax der Befehlsdatei vor. Gibt der Anwender übrigens hinter dem Namen der Befehlsdatei ein Fragezeichen an, so zeigt EXECUTE die Schablone des Scripts wie es von einem richtigen AmigaDOS-Befehl auch erwartet werden würde. Beispiel:

Die folgenden beiden Zeilen wurden unter dem Namen "bsp1" gespeichert:

.KEY wer/A,mit,ziel

Echo "<wer> kommt mit <mit\$niemand> ans <ziel\$Ziel>."

Die unterschiedlichen Argumente beim Starten der Datei mit EXECUTE erzeugen unterschiedliche Resultate:

1.Ram Disk:> execute bsp1 giuseppe magdalena
ziel=boot
giuseppe kommt mit magdalena ans boot.

Oben wurden alle drei Variablen mit einem Wert belegt, bei dem folgenden Start wurde nur ein Parameter übergeben. Deswegen hat EXECUTE die beiden anderen Variablen durch die nach dem Dollarzeichen angegebenen Voreinstellungen ersetzt:

1.Ram Disk:> execute bsp1 Wolfram
Wolfram kommt mit niemand ans Ziel.

Wie bei der Variablen wer/A können auch alle anderen mit den Optionen /A, /K, /S oder /F versehen werden, die die gleiche Bedeutung haben, wie die Argumente in der Schablone eines richtigen AmigaDOS-Befehles. Noch ein kleiner Hinweis: Achten Sie darauf, daß die .KEY-Zeile die erste Zeile der Script-Datei ist und die einzelnen

.DEF <arg>

Variablen nur durch Kommata voneinander getrennt werden, Leerzeichen dazwischen sind nicht erlaubt.

Soll einer Variablen für die ganze Script-Datei ein Vorgabewert gegeben werden und nicht nur an bestimmten Stellen (mit "\$"), so kann dieser Wert hier übergeben werden. Die Variable erhält diesen Wert aber nur, wenn Sie nicht vom Anwender anders belegt wurde. Beispiel:

.DEF mit "niemand"

.<leer>

In der ersten Zeile (und nur dort) kann ein Kommentar mit einem Punkt und mindestens einem Leerzeichen "." eingeleitet werden. Später in der Datei ist das Semikolon dafür reserviert.

Beginnt eine leere Kommentarzeile.

Alle diese Schlüsselwörter sollten gleich zu Beginn der Script-Datei eingesetzt werden, aber je Zeile immer nur eines (wie bei den eigentlichen Befehlen auch).

Bemerkung:

Zur Strukturierung von Script-Dateien stellt AmigaDOS viele Befehle zur Verfügung, etwa IF-ELSE-ENDIF oder SKIP-LAB-ENDSKIP bzw. QUIT oder FAILAT. Generell können alle AmigaDOS-Befehle - auch EXECUTE selbst - in einer Script-Datei verwendet werden.

Die Abarbeitung einer Befehlsdatei kann mit der Tastenkombination Ctrl-D oder dem entsprechenden Befehl "BREAK <proc> D" abgebrochen werden. Auf die Nummer des Prozesses, unter dem die Befehlsdatei bearbeitet wird, kann mit der Zeichenkette <\$\$> zugegriffen werden. Dies findet vor allem beim Erstellen eindeutiger Dateinamen für temporäre Zwischenspeicherung Anwendung. Eine Script-Datei kann auch als eigener Prozess laufen, indem "RUN" dem Befehl EXECUTE vorangestellt wird.

Siehe auch:

IF, SKIP, FAILAT, LAB, ECHO, RUN, QUIT, PROTECT

FAILAT

Syntax:

2.0/2.1/3.0: FAILAT [[RCLIM] <n>]

Schablone:

2.0/2.1/3.0: RCLIM/N

Pfad:

in die Shell integriert

Funktion:

Setzt den Fehlerwert, ab dem die Bearbeitung eines Scripts abgebrochen wird.

Beschreibung:

FAILAT ändert den Wert des Return-Codes, ab dem eine Script-Datei abgebrochen wird. Normalerweise gibt jeder AmigaDOS-Befehl über den sogenannten Return-Code dem System eine Mitteilung, ob er erfolgreich war, oder ob aufgrund irgendwelcher Fehler abgebrochen werden mußte.

FAILAT setzt nun für eine Script-Datei fest, wie groß der Fehler sein muß, damit die Bearbeitung beendet wird, denn im Normalfall hängt der Erfolg einer Befehlsdatei von den Einzelerfolgen der darin vorhandenen Befehle ab. Versagt ein Befehl, so ist ein Abbruch das kleinere Übel, bevor die weiteren Befehle versuchen mit wahrscheinlich falschen Daten weiterzuarbeiten.

Wird FAILAT kein Wert übergeben, so zeigt der Befehl die aktuelle Grenze für den Fehlerabbruch an.

Argument:

RCLIM/N	Eine Zahl, die festlegt, ab welchen Return-Code die Scriptdatei abgebrochen wird.
---------	---

Bemerkung:

Wenn ein Script von einer Shell gestartet wird, so wird die Fehlergrenze von der Shell übernommen, voreingestellt ist dort der Wert 10. Wird während der

Ausführung einer Befehlsdatei dieser Return-Code verändert, bleibt der neue Wert nur für diese Befehlsdatei gültig, nach Beendigung wird die voreingestellte Fehlergrenze 10 wieder verwendet.

Beispiel:

In den folgenden Zeilen wurde zunächst der aktuelle Wert der Shell geholt, anschließend dieser auf 30 erhöht und dann die Scriptdatei "test" ausgeführt, die nur drei FAILAT-Befehle enthält, der mittlere setzt die Grenze auf 20.

1.Ram Disk:> failat ; aktuellen Wert ermitteln

Fail limit: 10

1.Ram Disk:> failat 30 ; Grenze auf 30 erhöhen

1.Ram Disk:> failat ; Kontrolle: Hat Shell Wert übernommen?

Fail limit: 30

1.Ram Disk:> execute test ; Ja. Nun wird Script gestartet:

Fail limit: 30

Fail limit: 20

1.Ram Disk:> failat ; Welchen Wert hat die Shell hinterher?

Fail limit: 10

Siehe auch:

QUIT

FAULT

Syntax:

2.0/2.1/3.0: FAULT {<nn>}

Schablone:

2.0/2.1/3.0: /M/N

Pfad:

in die Shell integriert

Funktion:

Gibt den Text einer Fehlermeldung aus.

Beschreibung:

Es gibt Befehle, die eine Fehlermeldung ausgeben, aber nur in Form einer Zahl. Wenn Sie wissen wollen, wie dieser Fehler im Klartext aussieht, merken Sie sich die Nummer des Fehlers und geben dann die Nummer dem Befehl Fault als Argument. Fault kann mehrere Fehler gleichzeitig anzeigen. Die Fehlermeldungen werden in der jeweils eingestellten Sprache ausgegeben

Argument:

nn/M/N

Es können mehrere Fehlernummern übergeben werden.

Siehe auch:

WHY, Anhang Fehlermeldungen

FILENOTE

Syntax:

2.0/2.1/3.0: FILENOTE [FILE] <filepattern> [[COMMENT] <comments>]
[ALL] [QUIET]

Schablone:

2.0/2.1/3.0: FILE/A,COMMENT,ALL/S,QUIET/S

Pfad:

C:

Funktion:

Zu einer oder mehreren Dateien wird ein Kommentar angefügt.

Beschreibung:

Dies ist ein sehr nützlicher Befehl, mit dem man einzelne Dateien mit einem Kommentar versehen kann, der bis zu 79 Zeichen lang sein darf. Der Kommentar selbst erscheint nach der jeweiligen Datei bei der Ausführung des Befehles LIST. Wird nur ein Dateiname genannt und kein Kommentar angegeben, so wird der bereits vorhandene Kommentar gelöscht. Sollte der Kommentar Leerzeichen aufweisen, so ist der gesamte Text in Anführungszeichen zu setzen.

Argumente:

FILE/A Name oder Namensmuster der Datei(en), die mit einer Bemerkung versehen werden sollen.

COMMENT Die Bemerkung, die an die Dateien angehängt werden soll. Der Text darf nicht länger als 79 Zeichen sein. Escape-Sequenzen können auch verwendet werden, diese werden allerdings mehr als ein Zeichen in Anspruch nehmen, auch wenn sie in der Anzeige nur als einziges Zeichen dargestellt werden.

ALL/S Es werden mit dieser Option allen Dateien im aktuellen Verzeichnis mit dem angegebenen Kommentar versehen, oder der Kommentar wird überall gelöscht. Mit Namensmuster kann diese Option jedoch eingeschränkt werden.

QUIET/S Unterdrückt jegliche Ausgabe von FILENOTE.

Beispiele:

Füge an alle ".info"-Dateien den Kommentar "bildhübsch":

1.Ram Disk:Test> filenote #?.info "bildhübsch"

DataTypes.info...Done

Printers.info...Done

Keymaps.info...Done

Monitors.info...Done

DOSDrivers.info...Done

1.Ram Disk:Test> list

DataTypes.info	632	—	rw-d Today	23:28:23
: bildhübsch				
system-configuration	232	—	rw-d Today	23:28:23
serial.device	5412	—	rw-d Today	23:28:23
postscript_init.ps	5014	—	rw-d Today	23:28:23
printer.device	27420	—	rw-d Today	23:28:23
parallel.device	4272	—	rw-d Today	23:28:23
mfm.device	6684	—	rw-d Today	23:28:23
clipboard.device	6944	—	rw-d Today	23:28:23
Printers.info	632	—	rw-d Today	23:28:23
: bildhübsch				
Keymaps.info	632	—	rw-d Today	23:28:23
: bildhübsch				
Monitors.info	632	—	rw-d Today	23:28:23
: bildhübsch				
DOSDrivers.info	632	—	rw-d Today	23:28:23
: bildhübsch				
12 files - 75 blocks used				

Bemerkung:

Die Schutzbits (siehe PROTECT) werden durch FILENOTE nicht verändert.

Siehe auch:

LIST

FIXFONTS

Syntax:

2.0/2.1/3.0: FIXFONTS

Schablone:

2.0/2.1/3.0: -

Pfad:

Sys:System

Funktion:

Aktualisiert die “.font”-Dateien im logischen Gerät “FONTS:”.

Beschreibung:

FIXFONTS sollte immer dann angewandt werden, wenn der Inhalt des Verzeichnisses “FONTS:” geändert wurde. Es ändert die Einträge in den darin abgelegten “.font”-Dateien je nach dem Inhalt der einzelnen Unterverzeichnisse. Wird dieser Befehl nicht verwendet, wenn eine neue Schriftgröße bei einem Zeichensatz hinzugefügt oder eine andere gelöscht wurde, kann der Amiga verwirrt werden, da er in beiden Fällen die Zeichensätze nicht verwenden kann. Wurde eine Größe gelöscht, ist die Information selbst nicht mehr vorhanden, wurde eine hinzugefügt, ist die Größe noch nicht bekannt.

Bemerkung:

Findet FIXFONTS ein leeres Unterverzeichnis im logischen Gerät “FONTS:”, zu dem keine passende “.font”-Datei vorhanden ist, kann es zum Rechnerabsturz kommen.

Siehe auch:

ASSIGN

FKEY

Syntax:

2.0: FKEY [F<n>="text"] [SF<n>="text"] [CX_POPKEY="hotkey"]
[CX_POPUP=<YES/NO>] [CX_PRIORITY=<n>]

2.1/3.0: FKEY[CX_POPKEY="hotkey"] [CX_POPUP=<YES/NO>] [CX_PRIORITY=<n>]

Schablone:

2.0: KEY/M,CX_POPKEY/K,CX_POPUP/K,CX_PRIORITY/K/N

2.1/3.0: CX_POPKEY/K,CX_POPUP/K,CX_PRIORITY/K/N

Pfad:

Extras:Tools/Commodities

Funktion:

Belegt Funktionstasten mit Texten.

Beschreibung:

Dieser Befehl startet das gleichnamige Commodity-Programm, ändert die Tastenbelegungen oder die Priorität im Verhältnis zu den anderen aktiven Commodities. Es ist so beispielsweise möglich, häufig benutzte Amiga-DOS-Befehle auf Funktionstasten zu legen, die dann mit nur einem Tastendruck gestartet werden können. Die Funktionstasten können aber auch mit anderen Aktionen belegt werden, wie beispielsweise das Starten eines ARexx-Scripts oder Fenster-Funktionen. Wir empfehlen, einen Papierstreifen anzufertigen, der über die Funktionstasten gelegt werden kann, damit über jeder Taste zu sehen ist, welche Reaktion auf jede einzelne Taste erfolgt. Es gibt nichts schlimmeres als anstatt den Befehl "DIR #?" den Befehl "DELETE #?" aufzurufen. Generell hat "DELETE" hier eigentlich nichts verloren. Zum Löschen von Dateien und Verzeichnissen sollte man sich mehr Zeit nehmen - es lohnt sich auf die Dauer.

Argumente:

KEY/M

Nur für die Version 2.0 besteht die Möglichkeit, die Funktionstasten direkt mit dem Befehl "FKEY" mit

Text zu belegen. Die zu belegende Taste wird mit "Fn" (n = Nummer der Funktionstaste) bezeichnet und der gewünschte Text in Anführungszeichen zugewiesen. Sollte zusätzlich gleichzeitig die Shift-Taste gedrückt werden, so ist als Taste "SFn" einzutippen. Sollte unmittelbar im Anschluß an den Text ein Returnzeichen ausgegeben werden, ist der Text mit "\N" abzuschließen. Somit können in der Shell auch über die normale "Text-einfügen"-Funktion Befehle aufgerufen werden.

CX_POPUP/K Siehe Beschreibung des Befehles CLICKTOFRONT

CX_POPKEY/K Siehe Beschreibung des Befehles CLICKTOFRONT

CX_PRIORITY/K Siehe Beschreibung des Befehles CLICKTOFRONT

Bemerkung:

FKEY löst sich - wie auch die anderen Commodities - nicht von der Shell ab und sollte daher mit RUN gestartet werden. Seit der Workbench 2.1 wurden auch die Möglichkeiten des früheren Programmes "IHELP" in das Programm "FKEY" aufgenommen, das andere wurde von der Diskette genommen.

Durch das Blättersymbol kann für die aktuelle Taste der gewünschte Befehl ausgesucht werden. Es stehen folgende Möglichkeiten zur Verfügung (die jeweilige deutsche Bezeichnung steht in Klammern):

Cycle Window (Durch Fenster blättern) Holt das hinterste Fenster der Workbench in den Vordergrund - es werden jedoch nur Programmfenster nach vorne geholt, Schubladen bleiben hinten.

Cycle Screen (Durch Bildschirme blättern) Der hinterste Bildschirm wird in den Vordergrund geholt.

Insert Text (Text einfügen) Dies ist die Funktion, die schon bei Version 2.0 vorhanden war. Hier kann in das Gadget "Befehlsargumente" ein Text eingegeben werden, der

immer an der aktuellen Stelle eingefügt wird, sobald man die eingestellte Funktionstaste drückt.

- Enlarge Window** (Fenster vergrößern) Bringt das aktuelle Fenster auf die maximal mögliche Größe.
- Shrink Window** (Fenster verkleinern) Bringt das aktuelle Fenster auf die kleinst mögliche Größe.
- Toggle Window Size** (Fenstergröße umschalten) Diese Taste wirkt wie das Zoom-Gadget auf das aktuelle Fenster.
- Run Programm** (Programm starten) Kann jedes beliebige Programm starten. Der Programmname und alle notwendigen Parameter werden so in das Gadget "Befehlsargumente" eingegeben, als fände die Eingabe in einer Shell statt. Beachten Sie auch, daß der Amiga das entsprechende Programm finden kann - notfalls muß der komplette Pfad angegeben werden.
- Run ARexx Script** (ARexx-Script starten) Unter der Voraussetzung, daß der ARexx-Interpreter "ARexxMast" bereits interaktiv im Speicher ist, kann hier ein beliebiges ARexx-Script gestartet werden. Der Scriptname und die notwendigen Argumente werden auch hier im Gadget "Befehlsargumente" eingegeben.

Siehe auch:

AUTOPOINT, BLANKER, CLICKTOFRONT, EXCHANGE, IHELP, NOCAPSLOCK.

FONT

Syntax:

2.0/2.1/3.0: FONT [[FROM] <file>] [EDIT] [USE] [SAVE] [WORKBENCH]
[SCREEM] [SYSTEM]

Schablone:

2.0/2.1/3.0: FROM,EDIT/S,USE/S,SAVE/S,WORKBENCH/S,SCREEN/S,SYSTEM/S

Pfad:

Sys:Prefs

Funktion:

Startet das Preferences-Programm "Font", oder gibt die vom System verwendeten Zeichensätze aus.

Beschreibung:

Wird FONT ohne ein weiteres Argument oder mit dem Schlüsselwort "EDIT" aufgerufen, wird das Programm der Prefs-Schublade gestartet. Wird ein Dateiname angegeben und besitzt diese Datei Informationen zu den Zeichensatz-Voreinstellungen (sie wurde im Fonts-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert), können diese schon vorher definierten Einstellungen übernommen werden.

Argumente:

FROM Name einer Datei, die im Fonts-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert wurde. Was mit diesen bereits definierten Einstellungen passiert, wird mit den anschließenden Schlüsselwörtern festgelegt.

EDIT/S Startet das Programm "Fonts" der Prefs-Schublade, wie wenn das Icon auf der Workbench in der Schublade Prefs angeklickt worden wäre. Dies ist auch dann der Fall, wenn überhaupt kein Argument übergeben wird.

- USE/S** Wurde ein Dateiname einer Fonts-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen aktiviert. Beim Neustart des Rechners gelten jedoch wieder die alten Einstellungen.
- SAVE/S** Wurde ein Dateiname einer Fonts-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen gespeichert und damit bei jedem Neustart des Rechners aktiviert.
- WORKBENCH/S** Die Änderungen sollen nur für das Workbench-Fenster bzw. den Workbench-Screen vorgenommen werden. Diese Funktion kann von der Shell ausgeführt werden, ohne daß ein Requester fordert, alle Fenster außer der Workbench zu schließen.
- SCREEN/S** Die Änderungen werden für alle Bildschirme vorgenommen. Selbst wenn das Programm von der Shell gestartet wurde, fordert ein Requester, alle Fenster außer der Workbench zu schließen. Daher kann vorübergehend nicht mehr mit der Shell gearbeitet werden.
- SYSTEM/S** Die Änderungen sollen nur für System-Fenster (beispielsweise neue Shell-Fenster) vorgenommen werden. Diese Funktion kann von der Shell ausgeführt werden, ohne daß ein Requester fordert, alle Fenster außer der Workbench zu schließen.

FORMAT

Syntax:

- 2.0: FORMAT [DRIVE <drive>:] [NAME <name>] [FFS] [NOICONS] [QUICK]
- 2.1: FORMAT [DEVICE|DRIVE <drive>:] [NAME <name>] [OFS] [FFS]
 [INTL|INTERNATIONAL] [NOINTL|NOINTERNATIONAL]
 [NOICONS] [QUICK]
- 3.0: FORMAT [DEVICE|DRIVE <drive>:] [NAME <name>] [OFS] [FFS]
 [INTL|INTERNATIONAL] [NOINTL|NOINTERNATIONAL] [DIRCACHE]
 [NODIRCACHE] [NOICONS] [QUICK]

Schablone:

- 2.0: DRIVE/K/A, NAME/K/A, FFS/S, NOICONS/S, QUICK/S
- 2.1: DEVICE=DRIVE/K/A, NAME/K/A, OFS/S, FFS/S,
 INTL=INTERNATIONAL/S, NOINTL=NOINTERNATIONAL/S,
 NOICONS/S, QUICK/S
- 3.0: DEVICE=DRIVE/K/A, NAME/K/A, OFS/S, FFS/S,
 INTL=INTERNATIONAL/S, NOINTL=NOINTERNATIONAL/S,
 DIRCACHE/S, NODIRCACHE/S, NOICONS/S, QUICK/S

Pfad:

Sys:Systems

Funktion:

Formatiert eine Diskette.

Beschreibung:

Dieser Befehl formatiert eine neue Diskette, damit AmigaDOS oder CrossDOS darauf lesen oder schreiben kann. Der Befehl ist identisch zu dem Menüpunkt "Formatieren" der Workbench. Aber von der Shell aus hat der Anwender wesentlich mehr Möglichkeiten als bei einem Start über das Menü der Workbench. Die Grundsyntax ist folgende:

1> FORMAT DRIVE df0: NAME Empty

Die beiden Wörter DRIVE und NAME sind Schlüsselwörter und müssen unbedingt angegeben werden (vermutlich aus Gründen der Sicherheit vor versehentlichen Gebrauch des Befehl). Man kann nicht oft genug wiederholen, daß durch eine Formatierungen einer Diskette (im Gegensatz zum Befehl DELETE) sämtliche vorher darauf vorhandenen Daten vernichtet werden und zwar unwiderruflich. Der Befehl FORMAT kann zwar mit Ctrl-C abgebrochen werden, doch ist nach kürzester Zeit normalerweise das Löschen der Daten auch physikalisch soweit fortgeschritten, daß diese nicht mehr gerettet werden können. Dies gilt insbesondere auch für Festplatten, bei denen dieser Befehl auch angewandt werden könnte. Aber nur mit äußerer Vorsicht! Aus diesem Grund gibt es noch eine weitere Sicherheitsmeldung:

Insert disk to be formatted in drive df0: and press RETURN

Jetzt haben Sie die letzte Gelegenheit, die Formatierung abubrechen, in dem beispielsweise keine Diskette eingelegt und trotzdem RETURN gedrückt wird.

Wie der Befehl DISKCOPY gibt auch der Befehl FORMAT ständig Meldungen auf den Bildschirm aus, die ein starten mit dem Befehl RUN überaus schwierig gestalten. Sollten Sie auch bei der Ausführung dieses Befehls nicht auf das Multitasking verzichten wollen, so sollten Sie dem Befehl FORMAT eine eigene Shell zugestehen.

Sie können als Laufwerk physikalische Geräte wie "DFx:", "HDx:" (für Festplatten) oder auch "RAD:" und sogar "PCx:" angeben. Im letzten Fall wird die Diskette im MS-DOS-Format mit einer Kapazität von 720 KByte formatiert.

Argumente:

DRIVE/K/A

Dieses Schlüsselwort muß ausdrücklich angegeben werden und sagt dem Befehl FORMAT, in welchem Laufwerk die zu formatierende Diskette einliegt. Bei den Disketten sollten die Bezeichnungen "df0:" bis "df3:" - sofern vorhanden - verwendet werden, bei der Fesplatte empfiehlt sich die Angabe des Namens der zu formatie-

renden Partition. Sobald der Befehl begonnen hat, werden Daten unwiderruflich gelöscht - experimentieren Sie also nie mit der Festplatte!

NAME/K/A Auch dieses Schlüsselwort wird ausdrücklich von FORMAT verlangt. Es gibt an, wie die Diskette nach dem Formatieren genannt werden soll. Es gelten die allgemeinen Regeln für die Namensgebung unter AmigaDOS.

INTL/S Diese Option ist erst seit der Version 2.1 verfügbar und schaltet den Internationalen Modus ein. Hier wird die Diskette im sogenannte FS DOS-Format beschrieben. Dieser Schalter kann sowohl mit dem OFS (Old File System) als auch mit dem FFS zusammenarbeiten und korrigiert die alphabetische Reihenfolge der Verzeichnisse, in deren Namen sprachspezifische Umlaute in Groß- und Kleinbuchstaben vorkommen. Bei MS-DOS-Disketten kann diese Option nicht verwendet werden. Eine mit diesem Schalter formatierte Diskette kann mit Kickstart 1.2/1.3 nicht verwendet werden.

NOINTL/S Schaltet den internationalen Modus aus. Normalerweise braucht dieser Schalter nicht angegeben werden, da er voreingestellt ist.

DIRCACHE/S Diese seit der Version 3.0 vorhandene Option schaltet das sogenannte "Directory Caching" ein. Hierdurch wird die Geschwindigkeit, mit der Verzeichnisse gelesen werden können, so gesteigert, daß sie fast an Festplatten herankommen. Bei Festplatten wird nahezu die Geschwindigkeit der RAM:- Disk erreicht. Dafür wird beim Schreiben und Prüfen von Daten etwas mehr Zeit benötigt - ein Preis, den man jedoch in Kauf nehmen kann. Eine mit diesem Schalter formatierte Diskette kann mit Kickstart 1.2/1.3 nicht verwendet werden.

NODIRCACHE/S Schaltet den "Directory Caching"-Modus aus. Normalerweise braucht dieser Schalter nicht angegeben werden, da er voreingestellt ist.

- OFS/S** Mit diesem Schalter wird FORMAT dazu veranlaßt, das alte (Old File System) zu verwenden, damit die Disketten auch von Amigas mit Kickstart 1.2/1.3 verwendet werden können. Die Kapazität ist etwas geringer als Disketten im FFS-Format, auch sind die Zugriffszeiten etwas länger. Der Schalter OFS kann nicht in Verbindung mit dem Gerät "PCx:" (MS-DOS-Format) verwendet werden.
- FFS/S** Hier wird FORMAT gesagt, die Diskette soll mit dem schnelleren FFS beschrieben werden. Diese Diskette kann dann aber nicht mit Amigas mit Kickstart 1.2/1.3 verwendet werden. Der Schalter FFS kann nicht in Verbindung mit dem Gerät "PCx:" (MS-DOS-Format) verwendet werden.
- QUICK/S** Wie der Name schon sagt, wird eine Diskette hiermit im Schnelldurchgang formatiert. Dies ist jedoch nur dann möglich, wenn die Diskette schon vorher im AmigaDOS-Format beschrieben wurde. Es wird in diesem Fall lediglich das Inhaltsverzeichnis formatiert, alle anderen Spuren bleiben unberührt.
- NOICONS/S** Normalerweise schreibt der Befehl FORMAT nach erfolgter Formatierung das Verzeichnis "Trashcan" und das zugehörige Icon auf die frisch formatierte Diskette. Falls gewünscht, kann dies mit dem Schlüsselwort NOICONS unterbunden werden.

Bemerkung:

Gerade bei Amiga-Anwender die nur ein Diskettenlaufwerk besitzen, besteht die große Gefahr, daß Sie vergessen, die eigentlich zu formatierende Diskette einzulegen, so daß stattdessen die Workbench formatiert wird (denn von ihr muß der Befehl ja erst einmal geladen werden). Vergewissern Sie sich also bitte jedesmal vor dem Drücken der RETURN-Taste bei der Sicherheitsabfrage, daß wirklich die richtige Diskette im richtigen Laufwerk ist.

Es ist möglich, die Disketten mit dem FFS (Fast File System)-Format zu formatieren, wodurch die Diskettenzugriffe wesentlich beschleunigt werden

können. Bedenken Sie jedoch dabei, daß ältere Amigas diese Diskette nicht verarbeiten können, so daß ein Datenaustausch über dieses Format unter Kickstart 1.2/1.3 unmöglich ist.

Beispiel:

Ein relativ nervenschonender Versuch kann mit der resetfesten RAM-Disk RAD: gemacht werden (unter der Annahme daß alle darin vorhandenen Daten auch irgendwo auf einer Diskette vorhanden sein müssen und deswegen ein schmerzlicher Verlußt von Daten sehr unwahrscheinlich ist):

```
5.Workbench:Storage> format drive rad: name test1
```

Insert disk to be formatted in device RAD Press RETURN to begin formatting or CTRL-C to abort:

Verifying cylinder 79,0 to go Initializing disk...

Siehe auch:

DISKCOPY, INSTALL, RELABEL

GET

Syntax:

2.0/2.1/3.0: GET [NAME] <variable>

Schablone:

2.0/2.1/3.0: NAME/A

Pfad:

in die Shell integriert

Funktion:

Mit Get wird der Inhalt einer lokalen Variablen eingelesen.

Beschreibung:

Im Prinzip arbeitet der Befehl GET genauso wie sein Kollege GETENV. Der entscheidende Unterschied besteht darin, daß mit GET eine lokale Variable ausgelesen wird. Eine lokale Variable kann prinzipiell die gleichen Werte wie Umgebungsvariablen (auch globale Variable genannt) aufnehmen.

Die Umgebungsvariablen sind nur jedem Prozess zugänglich, da sie im logischen Gerät ENV: in Form einer Datei gespeichert sind, die lokalen Variablen sind aber auf einen Prozess festgelegt. Es ist möglich, sowohl eine lokale als auch eine globale Variable mit dem gleichen Namen zu definieren, da sie ja über zwei ganz unterschiedliche Befehle verwaltet werden.

Werden die Variablen jedoch über den Namen mit dem vorangestellten Dollar-Zeichen angesprochen (etwa in der Zeichenkette des ECHO-Befehls), so hat die lokale Variable Vorrang.

Argument:

NAME/A Name einer Variablen, deren Inhalt angezeigt wird.

Beispiel:

Erst wird eine lokale Variable definiert und anschließend ausgelesen:

1.Work:DOS3.0> set test1 “alles was recht ist”

1.Work:DOS3.0> get test1 alles was recht ist

Siehe auch:

SET, GETENV, SETENV

PAG Sandini

GETENV

Syntax:

2.0/2.1/3.0: GETENV [NAME] <variable>

Schablone:

2.0/2.1/3.0: NAME/A

Pfad:

in die Shell integriert

Funktion:

Mit Getenv wird der Inhalt einer globalen Variablen eingelesen.

Beschreibung:

Mit diesem Befehl kann der Wert einer globalen Variablen im Fenster angezeigt werden. Die globalen Variablen sind im logischen Gerät ENV: in Form von Dateien gespeichert und können deswegen in allen Prozessen angesprochen werden. Es ist möglich, sowohl eine lokale als auch eine globale Variable mit dem gleichen Namen zu definieren, da sie ja über zwei ganz unterschiedliche Befehle verwaltet werden. Werden die Variablen jedoch über den Namen mit dem vorangestellten Dollar-Zeichen angesprochen (etwa in der Zeichenkette des ECHO-Befehls), so hat die lokale Variable Vorrang.

Argument:

NAME/A Name einer Variablen, deren Inhalt angezeigt wird.

Beispiel:

Erst wird eine globale Variable definiert und anschließend ausgelesen:

```
1.Work:DOS3.0> setenv test1 "Das Böse ist immer und überall"  
1.Work:DOS3.0> getenv test1 Das Böse ist immer und überall
```

Siehe auch:

GET, SET, SETENV

GRAPHICDUMP

Syntax:

2.0/2.1/3.0: GRAPHICDUMP [TINY | SMALL | MEDIUM | LARGE
| <xPunkte>:<yPunkte>]

Schablone:

2.0/2.1/3.0: TINY/S, SMALL/S, MEDIUM/S, LARGE/S, x:y/N

Pfad:

Extras:Tools

Funktion:

Druckt den Bildschirm im Vordergrund aus.

Beschreibung:

Mit diesem Befehl ist es sehr einfach, ein Bildschirmfoto auszudrucken. Nach der Eingabe des Befehls wartet dieser noch ein paar Sekunden, bevor der Ausdruck beginnt, damit der Anwender Zeit hat, das gewünschte Fenster oder den gewünschten Bildschirm in den Vordergrund zu rücken. Mit den Größenoptionen kann die Größe des Ausdrucks bestimmt werden. Der Befehl verwendet die in den Voreinstellungen für den Drucker angegebenen Werte für seinen Ausdruck. Dies ist vorallem wichtig bei Farbdruckern.

Argumente:

- | | |
|----------|--|
| TINY/S | Diese Option gibt dem Ausdruck etwa 1/4 der maximalen Breite. Die Höhe wird der Bildschirmauflösung entsprechend angepaßt. |
| SMALL/S | Diese Option gibt dem Ausdruck etwa 1/2 der maximalen Breite. Die Höhe wird der Bildschirmauflösung entsprechend angepaßt. |
| MEDIUM/S | Diese Option gibt dem Ausdruck etwa 3/4 der maximalen Breite. Die Höhe wird der Bildschirmauflösung entsprechend angepaßt. |

LARGE/S

Diese Option gibt dem Ausdruck die mit dem Drucker erreichbare maximale Bildbreite. Die Höhe wird der Bildschirmauflösung entsprechend angepaßt.

<x>:<y>/N

Es können auch genaue Werte angegeben werden, wie groß das ausgedruckte Bild sein soll. Dabei müssen sowohl die Anzahl der Punkte in x- als auch in y-Richtung angegeben werden. Bei unverhältnismäßigen Werten kann das Bild stark verzerrt werden.

PAG Sandini

ICONTROL

Syntax:

2.0: ICONTROL [[FROM] <file>] [EDIT] [USE] [SAVE]

2.1/3.0: ICONTROL [[FROM] <file>] [EDIT] [USE] [SAVE]
[PUBSCREEN <screen>]

Schablone:

2.0: FROM,EDIT/S,USE/S,SAVE/S

2.1/3.0: FROM,EDIT/S,USE/S,SAVE/S,PUBSCREEN/K

Pfad:

SYS:Prefs

Funktion:

Es können von der Workbench verwendete Parameter angezeigt und verändert werden.

Beschreibung:

Wird ICONTROL ohne ein weiteres Argument oder mit dem Schlüsselwort "EDIT" aufgerufen, wird das Programm der Prefs-Schublade gestartet. Wird ein Dateiname angegeben und besitzt diese Datei Informationen zu den Voreinstellungen (sie wurde im ICONTROL-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert), können diese schon vorher definierten Einstellungen übernommen werden.

Argumente:

FROM Name einer Datei, die im ICONTROL-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert wurde. Was mit diesen bereits definierten Einstellungen passiert, wird mit den anschließenden Schlüsselwörtern festgelegt.

PAG Sandini

- EDIT/S** Startet das Programm "ICONTR0L" der Prefs-Schublade, wie wenn das Icon auf der Workbench in der Schublade Prefs angeklickt worden wäre. Dies ist auch dann der Fall, wenn überhaupt kein Argument übergeben wird.
- USE/S** Wurde ein Dateiname einer ICONTR0L-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen aktiviert. Beim Neustart des Rechners gelten jedoch wieder die alten Einstellungen.
- SAVE/S** Wurde ein Dateiname einer ICONTR0L-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen gespeichert und damit bei jedem Neustart des Rechners aktiviert.
- PUBSCREEN/K** Wird hier ein Name eines anderen Screens eingegeben, so wird das ICONTR0L-Fenster auf diesen Screen geöffnet.

ICONX

Syntax:

2.0/2.1/3.0: ICONX

Schablone:

2.0/2.1/3.0: -

Pfad:

C:

Funktion:

Ermöglicht das Starten von Befehlsdateien über Icons in der Workbench.

Beschreibung:

ICONX (der Name ist die Abkürzung für "ICON eXecute") kann nur von der Workbench aus verwendet werden, da es dafür vorgesehen ist, Script-Dateien von der Workbench aus zu starten. Dieser Befehl sollte in den Informationen zu einem Icon, die über den gleichnamigen Menüpunkt erreichbar sind, in das Gadget DEFAULT TOOL eingesetzt werden. Um einer Script-Datei ein Icon zu verpassen, muß ein Projekt-Piktogramm erstellt werden (mit IconEDIT) oder ein solches von einer anderen Datei kopiert und der Name angepaßt werden. Die Arbeitsweise von ICONX gleicht der des Befehles EXECUTE, mit dem Unterschied, daß keine Argumente übergeben werden können.

ICONX öffnet normalerweise selbstständig ein Fenster, in dem Ein- und Ausgaben für den Anwender durchgeführt werden können.

In den TOOL TYPES kennt ICONX noch ein paar interessante Schlüsselwörter:

WINDOW=

Hier kann eine Fensterdefinition der Form "con:x/y/breite/höhe/titel/zusätze" erstellt werden. Näheres entnehmen Sie bitte dem Kapitel zum Shellfenster am Anfang

des AmigaDOS-Teiles.

WAIT=

Die hier angegebene Zahl interpretiert ICONX als Anzahl der Sekunden, die das geöffnete Fenster nach Abschluß des Scripts noch geöffnet werden soll.

DELAY=

Die Funktion von Delay ist identisch zu WAIT, nur wird die hier angegebene Zahl in 1/50 Sekunden interpretiert.

STACK=

Sollte der Stapelspeicher, der auf 4096 Bytes voreingestellt ist, nicht ausreichen, so kann durch eine hinter STACK= anzugebende Zahl der verfügbare Stapelspeicher dem Bedarf entsprechend vergrößert werden.

Siehe auch:

EXECUTE

PAG Sandini

IF

Syntax:

2.0/2.1/3.0: IF [NOT] [WARN] [ERROR] [FAIL]
[<string> EQ|GT|GE <string>] [VAL] [EXISTS <file>]

Schablone:

2.0/2.1/3.0: NOT/S,WARN/S,ERROR/S,FAIL/S,,EQ/K,GT/K,GE/K,,VAL/S,
EXISTS/K

Pfad:

in die Shell integriert

Funktion:

In Script-Dateien können hier verschiedene Werte geprüft und die Befehlsausführung bedingt durchgeführt werden.

Beschreibung:

Dieser Befehl bildet das Herzstück der bedingten Bearbeitung von Befehlen in Scriptdateien, nämlich den Kopf mit der Abfrage der Bedingungen des IF-ELSE-ENDIF-Blocks. IF kann nur in Befehlsdateien verwendet werden und erzeugt in einer Shell eingegeben einen entsprechenden Hinweis. Der Grundaufbau eines IF-ESLE-ENDIF-Blocks ist folgender:

IF <Bedingung>

```
...           ;diese Befehle werden ausgeführt,  
...           ;falls die IF-Bedingung erfüllt ist.  
... ELSE  
...           ;diese Befehle werden ausgeführt,  
...           ;falls die IF-Bedingung nicht erfüllt ist.  
...
```

ENDIF

Als <Bedingung> kann eine der unten beschriebenen Möglichkeiten eingesetzt werden. Ist diese Bedingung erfüllt, werden die Befehle die zwischen der IF-Anweisung und dem Schlüsselwort ELSE (oder - wenn dieses nicht

vorhanden ist - der Anweisung ENDIF) ausgeführt. ELSE muß also nicht unbedingt ausgeführt werden, es kommt dann zum Einsatz, wenn bestimmte Befehle nur dann ausgeführt werden sollen, wenn die Bedingung nicht zutrifft. Diese Befehle werden dann zwischen den Befehlen ELSE und ENDIF eingefügt und dann ausgeführt, wenn die bei IF geforderte Bedingung nicht erfüllt ist.

Der Befehl ENDIF beendet den IF-ELSE-ENDIF-Block, d.h. nach dieser Zeile wird auf jedem Fall die Scriptdatei weiterbearbeitet - unabhängig von der Erfüllung der Bedingung - sofern kein größerer Fehler als mit FAILAT erlaubt entstanden ist. Es können mehrere IF-ELSE-ENDIF-Blöcke ineinander verschachtelt werden, jede IF-Anweisung muß aber ein zugehöriges ENDIF erhalten, andernfalls kommt der Befehl EXECUTE ins Schleudern.

IF ermöglicht auch einen direkten Zugriff auf Umgebungsvariablen durch voranstellen des Dollarzeichens.

Argumente: PAG Sandini

- NOT/S** Wird dieser Schalter zusätzlich zu einer Bedingung benutzt, so wird die Bedingung verneint, d.h. die Befehle unmittelbar nach IF werden dann ausgeführt, wenn die Bedingung nicht erfüllt wurde.
- WARN/S** Mit diesem Schlüsselwort wird getestet, ob der vorhergehende Befehl eine Warnung (Fehlercode 5) erzeugt hat. Diese Bedingung ist erfüllt, wenn eine Warnung erzeugt wurde.
- ERROR/S** Mit diesem Schlüsselwort wird getestet, ob der vorhergehende Befehl einen ERROR (Fehlercode 10) erzeugt hat. Diese Bedingung ist erfüllt, wenn ein ERROR erzeugt wurde. Achtung: Execute bricht in diesem Fall die Bearbeitung der gesamten Script-Datei ab, wenn nicht mit dem Befehl FAILAT die Fehlergrenze über 10 angehoben wurde.
- FAIL/S** Mit diesem Schlüsselwort wird getestet, ob der vorhergehende Befehl einen Fehler "FAIL" (Fehlercode 20) erzeugt hat. Diese Bedingung ist erfüllt, wenn ein FAIL erzeugt wurde.

Achtung: Execute bricht in diesem Fall die Bearbeitung der gesamten Script-Datei ab, wenn nicht mit dem Befehl FAILAT die Fehlergrenze über 20 angehoben wurde.

EQ/K

Dieses - wie auch die beiden nächsten - Schlüsselwort braucht einen vorhergehenden und einen nachfolgenden Wert. Dies kann eine Zeichenkette, eine Variable oder eine Zahl sein. Die Bedingung für den IF-Block ist dann erfüllt, wenn beide Werte identisch sind. Beispiele:

```
IF $zahl EQ 5
ECHO "Die Zahl hat den Wert 5"
ENDIF
```

;oder:

```
IF <arg> EQ ""
ECHO "Sie haben mir kein Argument übergeben, tut mir
leid!"
ENDIF
```

GT/K

Bei diesem Schlüsselwort ist die Bedingung dann erfüllt, wenn der erste Wert größer ist als der zweite. Hiermit können in Verbindung mit Umgebungsvariablen Schleifen mit Zähler konstruiert werden.

GE/K

Bei diesem Schlüsselwort ist die Bedingung dann erfüllt, wenn der erste Wert größer oder gleich dem zweiten ist. Hiermit können in Verbindung mit Umgebungsvariablen Schleifen mit Zähler konstruiert werden. Wenn Sie Bedingung "kleiner oder gleich" vermissen - versuchen Sie doch die Verbindung "NOT GT" und anstatt "kleiner" die Verbindung "NOT GE".

EXISTS/S

Mit diesem Schlüsselwort wird geprüft, ob eine bestimmte Datei oder ein bestimmtes Verzeichnis existiert. Dabei muß der entsprechende Name zusammen mit dem kompletten Pfad hinter dem Schlüsselwort EXISTS angegeben werden.

Die Bedingung ist dann erfüllt, wenn die Datei oder das Verzeichnis existiert. Es kann auch ein Gerätenamen auf seine Existenz hin geprüft werden, wird ein Gerät nicht gefunden, so wird der Anwender noch mit einem Requester aufgefordert, eine Diskette mit dem Namen des zu suchenden Gerätes einzulegen. Wird der Requester mit "Cancel" beantwortet, so ist die Bedingung nicht erfüllt, das Gerät wurde nicht gefunden. Beispielsweise erzeugt die Script-Datei

```
IF EXISTS test:
ECHO "test: ist vorhanden."
ELSE
ECHO "Ich kenne kein test:"
ENDIF
```

erst den Requester "Please insert Disk test: in any Drive", der mit "Cancel" beantwortet die Ausgabe

```
1> Execute Ram:test
```

```
Ich kenne kein test:
```

erzeugt. Wird jedoch test: definiert, so erscheint:

```
1> ASSIGN test: ram:
1> EXECUTE ram:test
test: ist vorhanden.
```

Es ist aber möglich, beim Prüfen der Existenz von logischen Geräten den Requester zu vermeiden, indem mit "ASSIGN <gerät> EXISTS" geprüft wird und das Ergebnis mit "IF WARN" abgefragt wird.

VAL/S

Dieses Schlüsselwort ist die Abkürzung des englischen Wortes "VALUE" und sagt der IF-Anweisung, daß die übergebenen Werte (in der Regel bei Variablen) numerisch geprüft werden. Dieses Schlüsselwort findet in Verbindung mit EQ, GT oder GE Anwendung.

Beispiele:

In folgender Script-Datei werden abhängig von der Schwere eines auftretenden Fehlers besondere Befehle abgearbeitet:


```
FAILT 25      ; damit EXECUTE nicht schon vorher aussteigt
...           ; hier steht ein kritischer Befehl
IF WARN
    ECHO "Es lief nicht ganz optimal!"
    SKIP Normal; trotzdem normal fortfahren
ENDIF
IF ERROR
    ECHO "Es trat ein Fehler auf!"
    SKIP Fehler; bestimmte Befehle einschieben
ENDIF
IF FAIL
    ECHO "Katastrophe! Katastrophe! Katastrophe!
    Katastrophe!"
    QUIT 25; sofortiger Abbruch der Befehlsdatei
ENDIF
LAB Fehler    ; hier beginnt die Fehlerbearbeitung
...
LAB Normal    ; evtl. Fehler behoben
ENDSKIP       ; und wieder normal weiter
...
```

Bemerkung:

Es ist sehr hilfreich, wenn bei einer bedingten Befehlsbearbeitung die Befehle zwischen den Schlüsselwörtern IF, ELSE und ENDIF um ein paar Zeichen eingerückt werden. Das erleichtert die Lesbarkeit und macht auch auf einen Blick erkennbar, welche Befehle wann ausgeführt werden. Gerade bei verschachtelten IF-Anweisungen gewinnt der Anwender hier an Übersichtlichkeit, die ihm bei einer Fehlersuche sehr schnell zu Gute kommt.

In älteren Versionen der Shell kann der IF-Befehl einen Fehler erzeugen, wenn in der gleichen Zeile mit einem Semikolon ein Kommentar hinzugefügt wurde.

Siehe auch:

ASSIGN, ELSE, ENDIF, ENDSKIP, EXECUTE, FAILAT, LAB, QUIT, SKIP,

IHELP

Syntax:

2.0: IHELP [CY_PRIORITY=<n>] [CYCLE=<"hotkey(s)">]
 [MAKEBIG=<"hotkey(s)">] [MAKESMALL=<"hotkey(s)">]
 [CYCLESSCREEN=<"hotkey(s)">] [ZIPWINDOW=<"hotkey(s)">]

2.1/3.0: Befehl durch FKEY ersetzt

Schablone:

2.0: CX_PRIORITY/K/N, CYCLE/K, MAKEBIG/K,
 MAKESMALL/K, CYCLESSCREEN/K, ZIPWINDOW/K

2.1/3.0: Befehl durch FKEY ersetzt

Pfad:

Extras:Tools/Commodity

PAG Sandini

Funktion:

Startet "IHELP" oder ändert die Einstellungen.

Beschreibung:

Wird IHELP ohne weiteren Argumente eingeben, so wird das gleichnamige Commodity-Programm geladen, wie wenn das entsprechende Icon in der Workbench angeklickt worden wäre. Mit Hilfe der Argumente kann aber verhindert werden, daß das Fenster erscheint, stattdessen werden die Einstellungen von der Shell aus definiert.

IHELP ermöglicht es, bestimmte Aktionen in der Workbench durch verschiedene Tastenkombinationen auszulösen, die normalerweise über Gadgets gesteuert werden. Als Tasten stehen folgende Möglichkeiten zur Verfügung:

Shift, Control, Alt, Funktionstasten, Help-Taste, alle druckbaren Zeichen

Bei Buchstaben ist zu beachten, daß IHELP die Groß- und Kleinschreibung unterscheidet, es sollten aber vor allem bei Verwendung von Buchstaben

noch andere Tasten als Bedingung gesetzt werden (z.B. "Control Alt a").

Argumente:

- CYLCE/K** (Durch Fenster blättern) Holt das hinterste Fenster der Workbench in den Vordergrund - es werden jedoch nur Programmfenster nach vorne geholt, Schubladen bleiben hinten. Voreingestellt ist diese Funktion auf der Funktionstaste F1.
- MAKEBIG/K** (Fenster vergrößern) Bringt das aktuelle Fenster auf die maximal mögliche Größe. Voreingestellt ist diese Funktion auf Taste F2.
- MAKESMALL/K** (Fenster verkleinern) Bringt das aktuelle Fenster auf die kleinst mögliche Größe. Voreingestellt ist diese Funktion auf Taste F3.
- CYCLEScreens/K** (Durch Bildschirme blättern) Der hinterste Bildschirm wird in den Vordergrund geholt. Voreingestellt ist diese Funktion auf Taste F4.
- ZIPWINDOW/K** (Fenstergröße umschalten) Diese Taste wirkt wie das Zoom-Gadget auf das aktuelle Fenster. Voreingestellt ist diese Funktion auf Taste F5.
- CX_PRIORITY/K** Siehe Beschreibung des Befehles CLICKTOFRONT

Bemerkung:

Wie auch alle anderen Commodore-Programme löst sich auch IHELP nicht vom startenden Prozess los, so daß das Programm mit RUN aufgerufen werden sollte, andernfalls wird die Shell stillgelegt, solange das Programm (auch im Hintergrund) läuft.

Siehe auch:

AUTOPOINT, BLANKER, CLICKTOFRONT, EXCHANGE, FKEY, NOCAPSLOCK.

INFO

Syntax:

2.0/2.1/3.0: INFO [[DEVICE] <device>:]

Schablone:

2.0/2.1/3.0: DEVICE

Pfad:

C:

Funktion:

Zeigt Informationen über phyikalische Gerät an.

Beschreibung:

Mit INFO erhält man eine Menge nützlicher Informationen zu den einzelnen Disketten und Festplattenpartitionen, sowie über die RAM:-Disk oder RAD:.

Die Informationen werden in einer Tabelle aufgeführt, in der jedes Gerät eine Zeile erhält. Die Bedeutung der einzelnen Spalten ist:

Unit	Der Name der physikalischen Einheit (Gerätename).
Size	Die formatierte Speicherkapazität des Gerätes.
Used	Der bereits belegte Speicher in Block (1 Block entspricht 512 Byte).
Free	Der noch freie Speicher in Block.
Full	Zeigt an, wieviel Prozent der gesamten Kapazität bereits belegt sind.
Errs	Anzahl der Fehler, die das Gerät seit dem Neustart des Amigas erzeugt hat.

Status Zeigt, ob die einliegende Diskette oder das Gerät schreibgeschützt ist oder nicht.

Name Name der Partition oder der eingelegten Diskette.

Volumes Available

Namen bekannter und verfügbarer Disketten und Festplattenpartitionen. Dies ist vergleichbar zu einem Teil der Anzeige des Befehls ASSIGN.

Argument:

DEVICE Sollen nur zu einem bestimmten Gerät die Informationen angezeigt werden, so muß der Gerätenamen als Argument übergeben werden.

Beispiele:

Der normale Befehl erzeugt:

1> info

PAG Sandini

Mounted disks:

Unit	Size	Used	Free	Full	Errs	Status	Name
RAD:	837K	2	1756	0%	0	Read/Write	RRD
PC0:	720K	1404	36	98%	0	Read Only	MSDOS5.0
RAM:	39K	39	0	100%	0	Read/Write	Ram Disk
HD0:	8267K	7448	9086	45%	0	Read/Write	Workbench
DF0:	Unreadable disk						
HD1:	107M	152241	68341	69%	0	Read/Write	Work

Volumes available:

MSDOS5.0 [Mounted]

RRD [Mounted]

Ram Disk [Mounted]

Work [Mounted]

Workbench [Mounted]

Nur Informationen über ein Gerät erwünscht:

1> info Workbench:

Mounted disks:

Unit	Size	Used	Free	Full	Errs	Status	Name
HD0:	8267K	7448	9086	45%	0	Read/Write	Workbench

Volumes available:

Workbench [Mounted]

Bemerkung:

Seit der Version 2.0 kann ein Gerät auch über den Namen der Diskette oder der Festplattenpartition eingegeben werden, wie obiges Beispiel zeigt. Und mit MS-DOS-Disketten hat INFO auch keine Schwierigkeiten, das "Unreadable Disk" im anderen Format soll nicht weiter stören.

Siehe auch:

ASSIGN

PAG Sandini

INITPRINTER

Syntax:

2.0/2.1/3.0: INITPRINTER

Schablone:

2.0/2.1/3.0: -

Pfad:

Extras:Tools

Funktion:

Sendet dem Drucker Initialisierungsbefehle, die durch das entsprechende Preferences-Programm vorgegeben wurden.

Beschreibung:

Der Drucker wird bei der ersten Benutzung über die Schnittstelle "PRT:" initialisiert, in dem die in dem Preferences-Programm "Printer" eingestellten Befehle dem Drucker geschickt werden. Sollte der Drucker hinterher zwischenzeitlich ausgeschaltet worden sein, oder ein Programm die Druckereinstellung geändert haben, so kann der Anfangszustand durch den Befehl INITPRINTER wieder hergestellt werden.

Bemerkung:

Kann ein Drucker nicht angesprochen werden, so wird der Befehl mit der Fehlermeldung FAIL (Fehlercode 20) abgebrochen, wenn der Requester "Printer Trouble..." mit "Cancel" beantwortet wurde.

INPUT

Syntax:

2.0: INPUT [[FROM] <file>] [EDIT] [USE] [SAVE]

2.1/3.0: INPUT [[FROM] <file>] [EDIT] [USE] [SAVE]
[PUBSCREEN=<screen>]

Schablone:

2.0: FROM,EDIT/S,USE/S,SAVE/S

2.1/3.0: FROM,EDIT/S,USE/S,SAVE/S,PUBSCREEN/K

Pfad:

SYS:Prefs

Funktion:

Startet das Voreinstellungsprogramm "INPUT" oder ändert die aktuellen Einstellungen.

Beschreibung:

Wird dieser Befehl ohne weitere Argumente oder mit dem Schlüsselwort EDIT gestartet, so erscheint das Fenster des Preferences-Programmes "Input".

Argumente:

FROM Name einer Datei, die im INPUT-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert wurde. Was mit diesen bereits definierten Einstellungen passiert, wird mit den anschließenden Schlüsselwörtern festgelegt.

EDIT/S Startet das Programm "INPUT" der Prefs-Schublade, wie wenn das Icon auf der Workbench in der Schublade Prefs angeklickt worden wäre. Dies ist auch dann der Fall, wenn überhaupt kein Argument übergeben wird.

- USE/S** Wurde ein Dateiname einer INPUT-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen aktiviert. Beim Neustart des Rechners gelten jedoch wieder die alten Einstellungen.
- SAVE/S** Wurde ein Dateiname einer INPUT-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen gespeichert und damit bei jedem Neustart des Rechners aktiviert.
- PUBSCREEN/K** Wird hier ein Name eines anderen Screens eingegeben, so wird das INPUT-Fenster auf diesen Screen geöffnet.

PAG Sandini

INSTALL

Syntax:

2.0/2.1/3.0: INSTALL [DRIVE] <DF0:|DF1:|DF2:|DF3:|CC0:> [NOBOOT] [CHECK]
 [FFS]

Schablone:

2.0/2.1/3.0: DRIVE/A, NOBOOT/S, CHECK/S, FFS/S

Pfad:

C:

Funktion:

Auf der Diskette im angegebenen Laufwerk wird der Bootblock geschrieben, gelöscht oder getestet.

Beschreibung: PAG Sandini

Die Hauptanwendung für diesen Befehl ist die Erstellung einer Boot-Diskette, d.h. es wird auf die angegebene Diskette die Information geschrieben, die dafür verantwortlich ist, daß der Amiga von dieser Diskette automatisch das Laden beginnt. Vor allem Anwender ohne Festplatte kennen den Effekt, daß nur bestimmte Disketten in der Lage sind, den Amiga zu starten, andere werden abgelehnt - auch wenn sie formatiert sind. Mit dem Befehl INSTALL kann jede Diskette zu einer Start-Diskette gemacht werden.

Argumente:

DRIVE/A	Es muß unbedingt der Gerätenamen des Laufwerkes angegeben werden deren Diskette bootfähig gemacht werden soll. Es ist auch möglich, PCMCIA-Karten bootfähig zu machen, der entsprechende Gerätenamen ist "CC0:".
---------	--

NOOBOOT/S	Hier wird eine bereits bootfähige Diskette für das Booten des Amigas unfähig gemacht. Dies ist dann sinnvoll, wenn die Diskette beispielsweise einen Virus enthält, oder die gespeicherten Daten für einen Start
-----------	--

nicht ausreichen (siehe dazu auch die Bemerkungen).

CHECK/S

Mit diesem Schlüsselwort wird geprüft, ob die eingelegte Diskette einen originalen Bootblock besitzt, bootunfähig ist oder gar von einem Boot-Virus infiziert ist.

FFS/S

Mit diesem Schlüsselwort wird ein spezieller FFS-Bootblock geschrieben. Diese Funktion ist seit der Version 2.1 wirkungslos aber aufgrund der Kompatibilität noch vorhanden.

Bemerkungen:

Auch wenn eine Diskette mit dem Befehl INSTALL bootfähig gemacht wurde, benötigt es noch viele weitere Dateien und Verzeichnisse, um daraus eine echte Start-Diskette zu machen. Werden außer dem Bootblock keine weiteren Dateien und Directories auf die Diskette gespeichert, so startet der Amiga, es ist aber nur das Shell-Fenster zu sehen. Jeglicher Versuch, einen AmigaDOS-Befehl einzugeben, wird mit der Meldung "unknown command" vereitelt.

INSTALL ist eine sehr einfache Form eines Virus-Killers. Der Befehl kann prüfen, ob der Bootblock verändert wurde (erster Hinweis auf einen Virus). Der Anwender kann ggf. den Bootblock neu schreiben lassen, wodurch ein Boot-Virus unschädlich gemacht werden kann. Vorsicht: Es gibt außer den Bootblock-Viren auch noch andere z.T. gemeinerer und unheilvollere Viren, INSTALL ist kein Allheilmittel!

Siehe auch:

FORMAT

IPREFS

Syntax:

2.0/2.1/3.0: IPREFS

Schablone:

2.0/2.1/3.0: -

Pfad:

C:

Funktion:

Dieses Programm startet einen Hintergrund-Prozess, der die Preferences-Voreinstellungen verwaltet. Er wird direkt von der originalen Startup-Sequence gestartet, liest dann die Voreinstellungen und setzt diese in die Tat um. Da vor diesem Befehl eigentlich keine Ausgabe erfolgen kann, da die Workbench noch "blind" ist, erzeugt IPREFS einen Requester, wenn ein Befehl in der Startup-Sequence schon vorher eine Meldung ausgeben wollte. In diesem Fall werden die festen Vorgaben des ROM verwendet.

Wird von einem Befehl eine Ausgabe erzeugt, reagiert IPREFS empfindlich: "close all windows except drawers". Daher ist es ratsam, in der Startup-Sequence keine der Ausgabeumleitungen ">NIL:" vor dem Befehl IPREFS zu entfernen.

JOIN

Syntax:

2.0/2.1/3.0: JOIN [FILE] <{oldfiles|pattern}> AS/TO <newfile>

Schablone:

2.0/2.1/3.0: FILE/M,AS=TO/K/A

Pfad:

C:

Funktion:

Verbindet mehrere Dateien zu einer.

Beschreibung:

BAG Sandini

Dieser Befehl ist sinnloser als der Anwender sich zunächst vorstellen kann. Mit JOIN können beliebig viele Dateien miteinander verknüpft und als eine einzige Datei gespeichert werden. Ist doch toll! - Schon, aber wofür werden die Dateien verknüpft?

JOIN macht keinen Unterschied, ob eine Datei ein ausführbares Programm, ein IFF-Bild oder andere Daten enthält, es verbindet alles, was gefordert wird. Die Vorstellung, daß mehrere Befehle mit JOIN verknüpft und dann nur noch über einen gemeinsamen Aufruf gestartet werden können, sollte schnell wieder vergessen werden. Die Befehle sind dann überhaupt nicht mehr aufrufbar, es kann zum Absturz des Rechners kommen und außerdem ermöglicht EXECUTE diese Verbindung von Befehlen sehr komfortabel.

Ein sinnvoller Einsatz ist nur bei reinen ASCII-Textdateien möglich. Hier können beispielsweise verschiedene Kapitel in eigenen Dateien gespeichert und bearbeitet werden und nach Fertigstellung zu einer großen Datei verknüpft werden - eine derartige Arbeitsweise habe ich bei diesem Buch angewandt (vermutlich sehr zum Leidwesen der Setzer - die mit einer Mammutdatei zu kämpfen haben).

Argumente:**FILE/M**

Eine Liste von Dateinamen oder Namensmuster, deren Inhalt zu einer neuen Datei zusammengefügt werden. Es ist möglich, soviele Dateien zu verbinden, wie Namen in eine AmigaDOS-Zeile passen (und mit den Jokerzeichen ist das eine gewaltige Menge).

AS=TO/K

Dieses Schlüsselwort muß angegeben werden und sagt JOIN, daß der folgende Dateiname die Zieldatei darstellt. Es dürfen keine Jokerzeichen verwendet werden.

Bemerkung:

Ganz so unnütz wie ich es oben dargestellt habe, ist JOIN nicht. Der Befehl kann vor allem in Script-Dateien wertvolle Dienste leisten.

PAG Sandini

LAB

Syntax:

2.0/2.1/3.0: LAB <marke>

Schablone:

2.0/2.1/3.0: /A

Pfad:

in die Shell integriert

Funktion:

Definiert eine Sprungmarke für den Befehl SKIP.

Beschreibung:

Dieser Befehl ist Teil der SKIP-LAB-ENDSKIP-Konstruktion für Script-Dateien und ist außerhalb von Befehlsdateien nicht anwendbar. Er definiert eine Sprungmarke, die mit einem passenden SKIP-Befehl angesprungen werden kann. Der Name der Sprungmarke kann beliebig lang sein, muß aber aus druckbaren Zeichen bestehen. In der Praxis reichen kurze prägnante Wörter aus. Eine Sprungmarke gilt nur für die jeweils ausgeführte Befehlsdatei, wird eine weitere Befehlsdatei aufgerufen, kann von dieser nicht auf die Sprungmarke der aufrufenden Script-Datei gesprungen werden.

Argument:

<marke>

Der Name einer Sprungmarke, kann beliebig festgelegt werden, wenn beachtet wird, daß keine nichtdruckbaren Zeichen enthalten sind. Werden Leerzeichen benötigt, so ist der gesamte Name in Anführungszeichen zu setzen.

Bemerkung:

Der Befehlsblock SKIP-LAB eignet sich vorzüglich, eine Konstruktion der Form "ON <eingabe> GOTO" zu verwirklichen, der beispielsweise in einem Bootmenü eingesetzt wird. Beispielsweise kann man mit folgendem Script zwischen "a" und "b" wählen.

In Verbindung mit den Befehlen TYPE, ECHO, SKIP uva. können sehr komfortable Bootmenüs erzeugt werden. Eine kleine Kostprobe finden Sie im Buch "Amiga Der Einstieg", das im gleichen Verlag erscheint.

Beispiel:

ECHO "Bitte *a*" oder *b*" eingeben!"

SKIP >NIL: ?

LAB a

ECHO "Du hast *a*" eingegeben!"

SKIP ende

LAB b

ECHO "Du hast *b*" eingegeben!"

SKIP ende

LAB ende

ECHO "Und Tschüß"

PAG Sandini

ENDSKIP

Wird dieses Script gestartet, kann man folgendes erreichen:

1.Work:DOS3.0> execute ram:test

Bitte "a" oder "b" eingeben!

a

Du hast "a" eingegeben!

Und Tschüß

1.Work:DOS3.0> execute ram:test

Bitte "a" oder "b" eingeben!

b

Du hast "b" eingegeben!

Und Tschüß

Siehe auch:

SKIP, ENDSKIP

LIST

Syntax:

2.0/2.1/3.0: LIST [{dirmuster}] [PIPAT <muster>] [KEYS] [DATES]
[NODATES] [TO <file>] [SUB <subname>] [SINCE <datum>]
[UPTO <datum>] [QUICK] [BLOCK] [NOHEAD] [FILES] [DIRS]
[LFORMAT <zeichenkette>] [ALL]

Schablone:

2.0/2.1/3.0:
DIR/M,P=PAT/K,KEYS/S,DATES/S,NODATES/S,TO/K,SUB/K,
SINCE/K,UPTO/K,QUICK/S,BLOCK/S,NOHEAD/S,FILES/S,
DIRS/S,LFORMAT/K,ALL/S

Pfad:

C:

PAG Sandini

Funktion:

Zeigt den Inhalt von Verzeichnissen und detaillierte Informationen zu den einzelnen Dateien.

Beschreibung:

LIST ist wohl einer der mächtigsten und wichtigsten Befehle des AmigaDOS, er hat viele Einsatzgebiete und ist dennoch sehr einfach zu beherrschen. Der Befehl kann sogar selbständig Befehlsdateien erstellen, wesentlich schneller als jede Sekretärin schreiben kann, dazu sehr effektiv und vor allem fehlerfrei.

Die einfachste Form ist der Befehl LIST ohne weitere Argumente, dann werden folgende Informationen angezeigt:

Name	Der Name der Datei oder des Verzeichnisses.
------	---

Größe	Die Größe einer Datei in Bytes, handelt es sich um ein Verzeichnis, so wird anstatt der Größe das Wort "Dir" angegeben.
-------	---

Schutzbits Es werden zu den Dateien oder Verzeichnissen die gesetzten Schutzbits angezeigt. Die Schutzbits können mit dem Befehl PROTECT geändert werden. Ist ein Bit gelöscht, so wird an seiner Stelle ein "-" gezeigt.

Datum, Zeit Hier wird das Datum und die Uhrzeit angezeigt, wann die Datei das letzte Mal geändert wurde. Diese Angaben sind jedoch nur dann aussagekräftig, wenn der Amiga über eine akkugepufferte Echtzeituhr verfügt.

Kommentar Ist bei einer Datei ein Kommentar eingefügt worden, so wird im Anschluß an die Zeile mit den Informationen zur Datei ein Doppelpunkt gesetzt und der Kommentar angezeigt.

In der letzten Zeile erscheint dann die Gesamtzahl der enthaltenen Dateien und Verzeichnisse, sowie der benötigte Speicherplatz in Blocks (512-Byte-Einheiten).

Beispiel:

PAG Sandini

1. Workbench:> LIST ram:

Directory "ram:" on Tuesday 19-Apr-94

Test 199 —rwed 21-May-78 23:39:21

: Dieses Script verdeutlicht die Verwendung des Skip-Befehles.

ENV Dir —rwed 21-May-78 22:30:25

Clipboards Dir —rwed 21-May-78 22:30:12

T Dir —rwed 21-May-78 22:30:23

1 file - 3 directories - 8 blocks used

Die Informationen, die LIST zur Verfügung stellt, sind sehr umfangreich, die Ausgabe kann aber durch die Schlüsselwörter sehr stark beeinflusst werden. Die einzelnen Schlüsselwörter und deren Bedeutung wird im Folgenden noch erklärt. Vorher sollten aber noch die Möglichkeiten im Zusammenhang mit der formatierten Ausgabe der Informationen bei Verwendung von LFORMAT genannt werden.

Mit dem Argument LFORMAT <zeichenkette> kann die Art der Informationsausgabe sehr stark beeinflusst werden. Es ist dadurch möglich,

Befehlsdateien automatisch erstellen zu lassen. Mit der Zeichenkette wird die Gestalt des Feldes, in dem die Informationen stehen, festgelegt. Ähnlich wie bei dem Befehl ECHO werden die Stellen, an dem LIST irgendwelche Informationen ausgeben soll, durch charakteristische Platzhalter mit einem einleitenden Prozentzeichen % festgelegt, der restliche Text wird übernommen.

Es stehen folgende Platzhalter zur Verfügung:

- %A An dieser Stelle werden die sogenannten "Attributes" - zu deutsch: "Schutzbits" eingesetzt.
- %B Hier wird die Größe der Datei in Blocks eingesetzt.
- %C Für den Kommentar ist dieser Platz reserviert.
- %D Natürlich kann auch das Datum hiermit platziert werden.
- %K Hier wird der sogenannte Key-Block angezeigt. Der Key-Block ist die Adresse einer Datei auf der Diskette.
- %L An dieser Stelle wird die Dateilänge in Bytes ausgegeben.
- %N Der Name der Datei ist wohl das wichtigste.
- %P Der Pfad der Datei darf natürlich nicht fehlen.
- %S Steht am Platz des Namens oder Pfadnamens der Datei, jenachdem, wie oft %S vorkommt (siehe dazu auch weiter unten).
- %T Dieser Platz ist für die Uhrzeit des letzten schreibenden Zugriffs auf diese Datei vorgesehen.

Seit der Version 2.1 gibt es noch zwei weitere Möglichkeiten:

- %E (Extension) Hier wird die sogenannte Namenserverweiterung (z.B. ".info") angezeigt.
- %M (Main) Und hier der eigentliche Name ohne Namenserverweiterung.

Diese Operatoren können noch weiter beeinflusst werden, in dem zwischen dem Prozentzeichen und dem Buchstaben noch weitere Zahlen eingefügt

werden. Beispielsweise kann der Operator %N erweitert werden: "%<xx>.<yy>N". Dabei steht <xx> bzw. <yy> jeweils für eine Zahl, erstere gibt die Breite des zur Verfügung stehenden Feldes an, die zweite die Anzahl der Zeichen in diesem Feld. Der Punkt und die zweite Zahl kann auch weggelassen werden. Der Text, der dann maximal <yy> Zeichen aufweist, wird in das <xx> Zeichen breite Feld rechtsbündig hineingeschrieben. Es gibt aber eine Einschränkung: als Feldgröße können keine Umgebungsvariablen verwendet werden.

Beispiel:

Mit der Folgenden Script-Datei namens "MyLIST" wurde die unten stehende Ausgabe erzeugt:

.key directory

```
LIST <directory> LFORMAT "Name: %10.10M Pfad: %20.20P Größe: %5L Erweiterung: %E"
```

Die Ausgabe sieht so aus:

1.Work:DOS3.0> MyLIST hd0:Storage/DOSDrivers

```
Name: AUX Pfad: hd0:Storage/DOSDrive Größe: 86 Erweiterung:
Name: AUX Pfad: hd0:Storage/DOSDrive Größe: 481 Erweiterung: info
Name: PC0 Pfad: hd0:Storage/DOSDrive Größe: 664 Erweiterung:
Name: PC0 Pfad: hd0:Storage/DOSDrive Größe: 492 Erweiterung: info
Name: PC1 Pfad: hd0:Storage/DOSDrive Größe: 670 Erweiterung:
Name: PC1 Pfad: hd0:Storage/DOSDrive Größe: 492 Erweiterung: info
Name: RAD Pfad: hd0:Storage/DOSDrive Größe: 549 Erweiterung:
Name: RAD Pfad: hd0:Storage/DOSDrive Größe: 481 Erweiterung: info
```

Um die Abwärtskompatibilität zu den älteren LIST-Versionen zu gewährleisten, wurde auch noch der Operand %S beibehalten. Dieser darf aber keine vorgegebene Feldgröße erhalten. Je nachdem, wie oft der Operator %S in der Zeichenkette auftaucht, wird er einmal durch den Dateinamen, ein andernmal durch den Pfadnamen ersetzt. Dabei gilt folgende Tabelle:

Anzahl %S	1. Mal	2. Mal	3. Mal	4. Mal
1	Dateiname			
2	Pfad	Dateiname		
3	Pfad	Dateiname	Dateiname	
4	Pfad	Dateiname	Pfad	Dateiname

Argumente:

- dirlmuster** Der Pfadname des Verzeichnisses, über dessen Inhalt Informationen gewünscht sind. Wird kein derartiges Argument übergeben, so werden die Daten zum aktuellen Verzeichnis gezeigt. Bei Verwendung von Muster werden nur Informationen zu den Dateien gezeigt, deren Namen zu dem angegebenen Muster passen. Im Pfadnamen können keine Jokerzeichen verwendet werden. Es können gleichzeitig auch mehrere Pfadnamen eingegeben werden. LIST behandelt eine mehrfache Eingabe, als wenn der Befehl hintereinander einzeln mit den jeweiligen Pfadnamen aufgerufen worden wäre (einzig die zwischenzeitliche Eingabeaufforderung der Shell entfällt).
- P=PAT/K** Hier kann zusätzlich zum Pfadnamen noch ein Namensmuster angegeben werden. Normalerweise wird dieses Schlüsselwort nicht benutzt, da die Namensmuster bereits beim Pfadnamen eingegeben werden können. In diesem Fall sollten aber nicht mit diesem Schlüsselwort weitere Muster eingegeben werden, wenngleich es möglich ist. Dann muß der Name zu beiden Mustern passen.
- DATES/S** Das Datum wird im Format TT-MMM-JJ und die Uhrzeit im Format HH:MM:SS ausgegeben. Dies entspricht der Voreinstellung, wenn nicht das Schlüsselwort QUICK angewandt wird.
- NODATES/S** Die Uhrzeit- und Datumsausgabe wird unterdrückt.
- TO <file>** Die Ausgabe wird in die angegebene Datei umgeleitet. Dieses Argument eignet sich in Verbindung mit dem Schlüsselwort LFORMAT vorzüglich zum Erstellen von Befehlsdateien. Wird dieses Schlüsselwort nicht verwendet, so erfolgt die Ausgabe in das Shell-Fenster.
- SUB/K** Hier kann ein Bruchstück des Namens der Dateien eingegeben werden, deren Informationen gewünscht werden. Vorhergehende oder nachfolgende Jokerzeichen sind nicht nötig.

- SINCE/K** Hier muß ein gültiges Datum oder ein Wort wie "yesterday", "today", "tomorrow" oder Wochentage eingegeben werden. Es werden nur die Dateien angezeigt, die bis einschließlich dieses Tages erstellt oder verändert wurden.
- UPTO/K** Hier muß ein gültiges Datum oder ein Wort wie "yesterday", "today", "tomorrow" oder Wochentage eingegeben werden. Es werden nur die Dateien angezeigt, die seit dem Tag erstellt oder verändert wurden.
- QUICK/S** Hier werden nur die Namen der Dateien und Verzeichnisse ohne jegliche Zusatzinformationen ausgegeben. Werden einzelne Informationen dennoch erwünscht (z.B. das Datum), so kann die Ausgabe wieder eingeschaltet werden (z.B. mit dem Schlüsselwort DATES).
- BLOCK/S** Normalerweise gibt LIST die Größe der Dateien in Bytes aus. Mit diesem Schlüsselwort wird die Größe jedoch in Block ausgegeben (1 Block entspricht 512 Byte).
- NOHEAD/S** Mit diesem Wort wird die Überschrift der LIST-Ausgabe unterdrückt. Alle übrigen Informationen erscheinen in gewohnter Weise.
- FILES/S** Es werden nur die Informationen zu den Dateien angezeigt, die vorhandenen Verzeichnisse werden verschwiegen.
- DIRS/S** Es werden nur die Informationen zu den Unterverzeichnissen angezeigt, die vorhandenen Dateien werden verschwiegen.
- LFORMAT/K** Hiermit wird das Ausgabeformat geändert. Näheres entnehmen Sie der obigen Beschreibung. LFORMAT setzt gleichzeitig auch die Option QUICK.
- ALL/S** Es werden mit diesem Argument auch die Informationen zu allen untergeordneten Verzeichnissen ausgegeben.

Bemerkungen:

Bei der Angabe eines Datums, muß auf die voreingestellte Sprache geachtet werden, ist beispielsweise Deutsch eingestellt, so muß eben "heute" statt today oder "Montag" an Stelle von "Monday" verwendet werden.

Normalerweise erfolgt die Ausgabe von LIST nicht alphabetisch sortiert. Mit dem Befehl SORT kann dies aber ermöglicht werden:

```
1> LIST >PIPE:a NOHEAD
```

```
1> SORT PIPE:A TO *
```

Beispiele:

In einem Verzeichnis und allen Unterverzeichnissen sollen möglichst schnell alle Dateien vor dem Löschen geschützt werden. Dies ist durch die Verwendung von nur zwei Befehlen möglich:

```
1> LFORMAT TestDIR TO RAM:BD LFORMAT "PROTECT %P%N wer" ALL
```

```
1> EXECUTE RAM:BD
```

Es soll ermittelt werden, wieviel Blöcke die einzelnen Dateien benötigen, und wo sie beginnen:

```
1.Work:DOS3.0> LIST df0: KEYS BLOCK
```

Directory "df0:" on Tuesday 19-Apr-94

dos1	[883]	87	—rwed Today	23:46:56
dos2	[1158]	173	—rwed Today	23:47:15
workbench	[1334]	305	—rwed Today	23:47:31
dos3	[972]	185	—rwed Today	23:56:32
dos4	[1646]	17	—rwed Today	22:56:24

5 files - 772 blocks used

Siehe auch:

DIR, PROTECT, FILENOTE, SETDATE

LOADWB

Syntax:

2.0: LOADWB [DELAY|DEBUG] [NEWPATH] [CLEANUP]

2.1/3.0: LOADWB [-DEBUG|DELAY] [CLEANUP] [NEWPATH]

Schablone:

2.0: DELAY/S,-DEBUG/S,NEWPATH/S,CLEANUP/S,

2.1/3.0: -DEBUG/S,DELAY/S,CLEANUP/S,NEWPATH/S

Pfad:

C:

Funktion:

Startet die Workbench

PAG Sandini

Beschreibung:

Dieser Befehl lädt und startet das Workbench-Programm. LOADWB selbst ist aber nicht das gesamte Programm, es ist lediglich der Befehl, mit dem die Workbench gestartet wird. Ein sehr großer Teil der Arbeit wird vom Kickstart (im ROM befindliches Betriebssystem) übernommen. In der originalen Startup-Sequence wird dieser Befehl aufgerufen, sodaß die Workbench automatisch erscheint, auch wenn sie nicht benutzt wird.

Argumente:

-DEBUG/S

Mit diesem Schlüsselwort wird das speziell für Entwickler vorgesehene Menü "Debug" in das Menü der Workbench aufgenommen. Darin befinden sich die beiden Unterpunkte "DEBUG" und "FlushLibs". Mit dem Menüpunkt DEBUG wird der Debugger "ROM Wack" gestartet, der detailliertere Fehlermeldungen an die serielle Schnittstelle sendet. Anmerkung: Mein Versuch, diesen Menüpunkt auszuwählen, wurde mit einem Absturz bestraft. "FlushLibs" entfernt nicht mehr benötigte

Libraries, sofern sie unnötig RAM-Speicher verschwenden. Diese Option kann nicht mit DELAY kombiniert werden.

- DELAY/S** Mit diesem Schlüsselwort wird das Öffnen der Workbench um drei Sekunden verzögert. Dahinter steckt der Gedanke, daß andere Befehle (etwa in einer Befehlsdatei) vorher noch ihre Diskettenaktionen abschließen können. Dadurch kann ein unnötig häufiges Hin- und Herfahren des Schreib/Lese-Kopfes vermieden werden, was wiederum das Laufwerk schont.
- NEWPATH/S** Mit diesem Schlüsselwort übernimmt die Workbench den Suchpfad von der Shell, damit später von der Workbench gestartete Shell wieder diesen Pfad verwenden können.
- CLEANUP/S** Jedes Icon hat seinen festen Stammplatz im Workbench-Fenster. Es kann aber passieren, daß dieser schon beim Starten der Workbench besetzt ist, so daß unter Umständen ein mehr oder minder großes Chaos entstehen kann. Mit diesem Schlüsselwort werden gleich zu Beginn alle darzustellenden Icons geordnet, bevor sie in der Workbench gezeichnet werden.

LOCALE

Syntax:

2.0: Befehl nicht implementiert

2.1/3.0: LOCALE [[FROM] <file>] [EDIT] [USE] [SAVE]
[PUBSCREEN <screen>]

Schablone:

2.0: Befehl nicht implementiert

2.1/3.0: FROM,EDIT/S,USE/S,SAVE/S,PUBSCREEN/K

Pfad:

SYS:Prefs

Funktion:

Startet das Voreinstellungsprogramm "Locale" oder ändert die aktuellen Einstellungen.

Beschreibung:

Wird dieser Befehl ohne weitere Argumente oder mit dem Schlüsselwort EDIT gestartet, so erscheint das Fenster des Preferences-Programmes "LOCALE".

Argumente:

FROM Name einer Datei, die im LOCALE-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert wurde. Was mit diesen bereits definierten Einstellungen passiert, wird mit den anschließenden Schlüsselwörtern festgelegt.

EDIT/S Startet das Programm "LOCALE" der Prefs-Schublade, wie wenn das Icon auf der Workbench in der Schublade Prefs angeklickt worden wäre. Dies ist auch dann der Fall, wenn überhaupt kein Argument übergeben wird.

USE/S	Wurde ein Dateiname einer LOCALE-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen aktiviert. Beim Neustart des Rechners gelten jedoch wieder die alten Einstellungen.
SAVE/S	Wurde ein Dateiname einer LOCALE-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen gespeichert und damit bei jedem Neustart des Rechners aktiviert.
PUBSCREEN/K	Wird hier ein Name eines anderen Screens eingegeben, so wird das LOCALE-Fenster auf diesen Screen geöffnet.

PAG Sandini

LOCK

Syntax:

2.0/2.1/3.0: LOCK [DRIVE] <drive>: [ON|OFF] [<passkey>]

Schablone:

2.0/2.1/3.0: DRIVE/A, ON/S, OFF/S, PASSKEY

Pfad:

C:

Funktion:

Sperrt ein Gerät für den schreibenden Zugriff.

Beschreibung:

LOCK wurde als Teil des "Fast File System"-Pakets eingeführt und sollte den unbefugten schreibenden Zugriff auf FFS Festplatten-Partitionen verhindern.

Solange eine Partition gesperrt ist, kann kein Programm darauf schreiben, bis die Sperre aufgehoben wird. Dazu müssen aber ein paar Dinge erläutert werden. LOCK stellt lediglich einen softwareseitigen Schreibschutz dar, d.h. es kann nur solange vor unbefugten Zugriff schützen, bis der Amiga neu gebootet wird, oder der Schreibschutz ausdrücklich entfernt wird. Er hindert auch nur AmigaDOS daran, auf das geschützte Gerät zu schreiben. Auch wenn die meisten Programme über das AmigaDOS auf die Daten und Dateien der Festplatte zugreifen, gibt es wenige Programme, die auf eigene Faust arbeiten, für diese ist dieser Schreibschutz unwirksam.

ACHTUNG: LOCK bietet keinerlei Schutz vor Virenbefall oder anderen schreibende Zugriffe, die nicht über AmigaDOS durchgeführt werden.

Seit der Version 2.0 wurde LOCK stark verbessert, der Befehl läuft jetzt auf allen Geräten, das FFS ist kein Muß mehr. Erwähnenswert ist, daß LOCK die Geräte sperrt und nicht die austauschbaren Disketten. Wird beispielsweise das Gerät df0: gesperrt, so kann auch nach einem Diskettenwechsel die in

df0: einliegende Diskette solange nicht beschrieben werden, bis der Schreibschutz definitiv aufgehoben ist oder der Rechner neu gestartet wurde. Es können auch logische Geräte gesperrt werden, ein Schreiben ist jedoch selbst auf geschützten logischen Geräten möglich, wenn nicht das Gerät selbst, sondern der entsprechende Pfad als Ziel angegeben wird.

Argumente:

DRIVE/A Hier muß der Name des zu sperrenden Gerätes angegeben werden. Festplattenpartitionen können auch am eigenen Namen genannt werden.

ON/S Sperrt das angegebene Gerät für den schreibenden Zugriff.

OFF/S Hebt die Sperre auf.

PASSKEY Hier kann ein Paßwort eingegeben werden. Wurde eines eingegeben, kann der Schreibschutz nur dann wieder aufgehoben werden, wenn das exakt gleiche Paßwort wieder eingetippt wird. Die Länge des Paßwortes ist auf vier Zeichen beschränkt, die Groß- und Klein-Schreibung wird jedoch berücksichtigt, Kontrollzeichen sind ebenfalls möglich.

Beispiel:

```
1.Work:DOS3.0> lock df0: on fd
df0: locked
```

```
1.Work:DOS3.0> copy #?4 to df0:
```

```
Can't open dos4 for output - disk is write-protected
```

```
1.Work:DOS3.0> dir df0:
```

```
dos1                dos2
```

```
dos3                dos4
```

```
workbench
```

Bemerkung:

LOCK sperrt nur den schreibenden Zugriff, Daten von dem schreibgeschützten Gerät zu lesen ist genauso möglich wie von einer hardwareseitig schreibgeschützten Diskette.

MAGTAPE

Syntax:

2.0/2.1/3.0: MAGTAPE [DEVICE <devicename>] [UNIT <n>] [RET | RETENSION]
[REW | REWIND] [SKIP <n>]

Schablone:

2.0/2.1/3.0: DEVICE/K, UNIT/K/N, RET=RETENSION/S, REW=REWIND/S,
SKIP/K/N

Pfad:

C:

Funktion:

Dieser Befehl steuert Band-Laufwerke (sogenannte "Streamer") an.

Beschreibung: PAG Sandini

Der Befehl MAGTYPE ermöglicht das gezielte Spulen von Magnetbändern. Es kann jeden SCSI-Streamer ansprechen, wenn der von Commodore genormte Befehlssatz unterstützt wird. Dazu muß nur der Name des Gerätetreibers hinter dem Schlüsselwort DEVICE genannt werden.

Argumente:

- | | |
|-----------------|---|
| DEVICE/K | Hier kann der Name des Gerätetreibers angegeben werden, über den das Band angesteuert werden soll. |
| UNIT/K/N | Hier wird die Nummer des Gerätes angegeben, sofern sie nicht dem voreingestellten Wert 4 entspricht. |
| RET=RETENSION/S | Durch diesen Schalter wird das Band zuerst bis zum Ende vorgespult und anschließend komplett zurückgespult. Der Hintergrund ist, daß das Band korrekt aufgewickelt sein sollte, bevor es gelagert wird. |

REW=REWIND/S

Das Band wird an den Anfang zurückgespult.

SKIP/K/N

Die auf das Schlüsselwort SKIP folgende Zahl gibt an, wieviel Segmente übersprungen werden sollen. Hiermit ist es möglich mehr als nur einen Eintrag je Band zu speichern und wieder auszulesen.

Bemerkung:

Da mir die entsprechende Hardware fehlt, bin ich bei der Beschreibung dieses Befehls auf andere Literatur angewiesen. Ich konnte diesen Befehl nicht testen, so daß auch die Beschreibung sehr knapp und z.T. holprig ausfiel. Da aber wohl die wenigsten diesen Befehl je benutzen werden und beim Kauf eines derartigen Gerätes andere Software und weitere Unterlagen mitgeliefert werden sollten, bin ich der Meinung, daß die obige Beschreibung vollkommen ausreicht.

PAG Sandini

MAKEDIR

Syntax:

2.0/2.1/3.0: MAKEDIR [NAME] <{name}>

Schablone:

2.0/2.1/3.0: NAME/M

Pfad:

C:

Funktion:

Erstellt neue Unterverzeichnisse.

Beschreibung:

MAKEDIR erzeugt in einem bereits vorhandenen Verzeichnis weitere Unterverzeichnisse, jedoch keine ".info"-Dateien und somit keine Symbole für die Workbench. Wenn man jedoch auf die Bildchen nicht verzichten will, geht es schneller, wenn eine leere Schublade auf der Workbench mit dem Menüpunkt "Copy" kopiert und anschließend umbenannt wird. Das Erzeugen eines neuen Verzeichnisses ist denkbar einfach, es braucht nur der Name des neuen Directory eingegeben werden und, wenn es nicht im aktuellen Verzeichnis entstehen soll, auch der Pfadname dazu.

Es gibt jedoch ein paar Einschränkungen, die seit der ersten Version des AmigaDOS noch nicht beseitigt sind. So kann MAKEDIR immer nur Unterverzeichnisse einer Hierarchiestufe erzeugen, darin aber nicht noch weitere Unterverzeichnisse. Wenn in der Ram:-Disk beispielsweise das Directory Test1 und darin eines mit den Namen Test2 entstehen soll, muß der Befehl zweimal aufgerufen werden, ein einzelner Aufruf produziert eine Fehlermeldung:

```
1.Work:DOS3.0> makedir ram:Test1/Test2
```

```
Can't create directory ram:Test1/Test2
```

```
object not found
```

```
1.Work:DOS3.0> makedir ram:Test1
```

```
1.Work:DOS3.0> makedir ram:Test1/Test2
```


Es gibt jedoch eine Alternative:

```
1> MAKDIR ram:Test1 ram:Test1/Test2
```

Es können als mit einem Befehl durchaus mehrere Verzeichnisse angelegt werden, nur müssen diese jeweils einzeln mit vollständigen Pfad genannt werden. (Anmerkung: Der Befehl COPY ist in der Lage, fehlende Unterverzeichnisse gleich welcher Ebene in einem Durchgang zu erzeugen.) Solltes eines oder gar mehrere der neu anzulegenden Verzeichnisse bereits vorhanden sein, so erzeugt MAKEDIR zwar einen entsprechenden Fehler (Code 10), führt aber die Befehlszeile - soweit möglich - weiter aus. Der Befehl versucht also, möglichst viele der geforderten Verzeichnisse zu erzeugen.

Bei den verwendeten Verzeichnisnamen gelten verständlicherweise die Regeln für Verzeichnis- und Dateinamen.

Argumente:

NAME/M Der Namen eines neuen Verzeichnisses. Soll das Verzeichnis nicht im aktuellen Verzeichnis angelegt werden, so ist der Pfadname ebenfalls anzugeben. Es können mit einem Befehlsaufruf von MAKEDIR beliebig viele Verzeichnisse erstellt werden, allerdings immer nur eines je Verzeichnisebene. Auf die Regeln der Namensgebung unter AmigaDOS muß geachtet werden.

Siehe auch:

ASSIGN, CD

MAKELINK

Syntax:

2.0/2.1/3.0: MAKELINK [FROM] <file> [TO] <file> [HARD] [FORCE]

Schablone:

2.0/2.1/3.0: FROM/A, TO/A, HARD/S, FORCE/S

Pfad:

C:

Funktion:

Verbindet zwei Dateien.

Beschreibung:

Dieser etwas merkwürdige Befehl erzeugt eine Datei, die auf eine andere Datei zeigt, welche ausführbar sein muß. Wird der Name der FROM-Datei eingegeben, so wird die TO-Datei ausgeführt. Dies ist auch dann der Fall, wenn andere Programme die Datei aufrufen.

Die Verbindungen achten nicht auf Veränderungen im entsprechenden Pfad, so daß eine Verbindung nach einer Änderung der Verzeichnisstruktur nicht mehr erkannt wird. Gegenwärtig können nur sogenannte "Hard Links" erstellt werden, ein solcher entsteht, wenn zwei Dateien auf dem gleichen Gerät miteinander verbunden werden. Im Gegensatz dazu ist auch in späteren Versionen von der Möglichkeit, Soft Links zu erstellen, die Rede, die dann geräteübergreifend Dateien verbinden sollten.

Argumente:

FROM/A Der Name der Quell-Datei, die an die TO-Datei gebunden wird. Diese Datei wird durch LINK erstellt, darf also vorher noch nicht vorhanden sein.

TO/A Der Name der Ziel-Datei, die anstelle der FROM-Datei aufgerufen wird.

- HARD/S** Eigentlich sollte MAKELINK Soft-Links erstellen, wenn dieses Schlüsselwort nicht angegeben ist und erst dieses Schlüsselwort sollte MAKELINK zur Herstellung eines Hard-Links bewegen. Da aber derzeit noch keine Soft-Links möglich sind, werden immer Hard-Links erstellt, dieser Schalter ist derzeit noch ohne Bedeutung.
- FORCE/S** Hiermit kann MAKELINK gezwungen werden, Verbindungen zwischen Dateien in verschiedenen Verzeichnissen zu erstellen.

Beispiele:

In diesem wohl nicht sehr einsichtigen Fall sind Beispiele lehrreicher:

Zunächst erstellen wir eine Zielfeile anschließend eine Kette von Links. In der letzten Zeile wurde versucht, eine Endlosschleife zu ziehen, was jedoch unterbunden wurde:

```
1.Ram Disk:> echo >Test "Dies ist meine Ziel-Datei"
1.Ram Disk:> makelink lnk1 test
1.Ram Disk:> makelink lnk2 lnk1
1.Ram Disk:> makelink lnk3 lnk2
1.Ram Disk:> makelink lnk4 lnk3
1.Ram Disk:> makelink test lnk4
object already exists
```

Wenn wir den Verzeichnisinhalte mit DIR anschauen, erscheint hinter den Link-Dateien die Zeichenfolge <hl> als Zeichen, daß diese Datei mit einer anderen über einen Hard-Link verbunden ist:

```
1.Ram Disk:> dir
lnk1 <hl>          lnk2 <hl>
lnk3 <hl>          lnk4 <hl>
Test
```

Mit dem Befehl List wird dies jedoch nicht dargestellt, stattdessen hat jede Link-Datei die Größe der Datei, auf die sie sich eigentlich bezieht.

1.Ram Disk:> list

Ink4	26	—rwed Today	22:50:30
Ink3	26	—rwed Today	22:50:30
Ink2	26	—rwed Today	22:50:30
Ink1	26	—rwed Today	22:50:30
Test	26	—rwed Today	22:50:30

5 files - 0 directories - 16 blocks used

Wenn wird die letzte Datei der Kette aufrufen, wird die erste angesprochen:

1.Ram Disk:> type Ink4 Dies ist meine Ziel-Datei

Was geschieht, wenn ein Glied in der Kette gelöscht wird?

1.Ram Disk:> delete Ink3

Ink3 Deleted

1.Ram Disk:> delete Ink2

Ink2 Deleted

1.Ram Disk:> type Ink4

Dies ist meine Ziel-Datei

PAG Sandini

Die Zuweisung bleibt trotzdem bestehen, es werden in diesem Fall die beiden nachbarten Glieder des zu löschenden Kettengliedes miteinander verknüpft. Wird jedoch die Zieldatei verändert, so wird die Verbindung zum nächsten Glied aufgelöst und dieses Glied in eine richtige Datei mit dem gleichen Inhalt der alten Zieldatei umgewandelt.

MEMACS

Syntax:

2.0/2.1/3.0: MEMACS [<file>] [GOTO <n>] [OPT W]

Schablone:

2.0/2.1/3.0: FILE, GOTO/K/N, OPT W/S

Pfad:

Extras:Tools

Funktion:

Startet den Texteditor MEmacs.

Beschreibung:

MEMACS ist eine Abkürzung für MicroEmacs (von David Conroy), der beste Texteditor, der für den Amiga mitgeliefert wurde. Dieser Editor kann für praktisch sämtliche Zwecke zur Bearbeitung von ASCII-Texten (wie beispielsweise Befehlsdateien o.ä.) verwendet werden. Dieser Editor ist sehr viel umfangreicher und anwendungsfreundlicher als ED, ein Vergleich mit EDIT erübrigt sich. (Dieses Manuskript habe ich mit einem nahen Verwandten von MEmacs erstellt: M2Emacs - ein Editor, der im Programmierpaket M2Amiga (Modula-2) enthalten ist.)

Wir beschränken uns hier jedoch vorerst auf die Beschreibung der Befehlszeile, der Beschreibung des Editors selbst ist ein eigenes Kapitel gewidmet. der Editor kann durch den Befehl MEMACS ohne ein weiteres Argument gestartet werden. Es können aber auch folgende Argumente verwendet werden:

Argumente: <file> Der Name einer Text-Datei, die mit MEMACS bearbeitet werden soll. Es muß aber nicht unbedingt angegeben werden. Ist eine solche Datei vorhanden, wird sie nach dem Programmstart automatisch geladen.

- GOTO/K/N** Wird eine Datei geladen, so kann sie mit dem Argument GOTO <n> ab der <n>-ten Zeile angezeigt werden.
- OPT W/S** Wird dieser Schalter übergeben, öffnet MEMACS ein Fenster in der Workbench und nicht wie sonst üblich einen eigenen Screen. Dies ist dann sehr hilfreich, wenn Speicher Mangelware ist, andernfalls sollte diese Option nicht verwendet werden.

Siehe auch:

ED, EDIT

PAG Sandini

MORE

Syntax:

2.0/2.1/3.0: MORE [<file>|pipename>]

Schablone:

2.0/2.1/3.0: <name>

Pfad:

Sys:Utilities

Funktion:

Zeigt den Inhalt einer ASCII-Datei an.

Beschreibung:

MORE ist der Dateianzeiger des Amigas, der jede beliebige ASCII-Textdatei ähnlich dem Befehl TYPE darstellen kann, jedoch wesentlich komfortabler. Dieser Befehl wird oft verwendet, um die bekannten "Read.Me"-Dateien der PD-Disketten oder anderer Programme anzuzeigen. Normalerweise wird in der Befehlszeile der Namen der zu lesenden Datei angegeben. Ist dies nicht der Fall, so fordert MORE den Anwender dazu auf.

MORE öffnet ein eigenes Fenster und stört dadurch den Inhalt des Shell-Fensters, wenn er mit dem Befehl RUN gestartet wurde. In diesem Fall erscheint auch seit der Version 2.0 ein Filerequester, mit dem die anzuzeigende Textdatei bequem ausgewählt werden kann. Wird MORE ohne RUN gestartet, so erfolgt die Anzeige im aktuellen Shell-Fenster.

MORE kann seine Datei auch vom logischen Gerät PIPE: beziehen. In diesem Fall ist es empfehlenswert, MORE durch den Befehl RUN als eigenständigen Prozess zu starten, da es mitunter länger dauert, bis in PIPE: irgendwelche Daten anliegen.

Der Textanzeiger MORE besitzt außer einem optionalen Dateinamen keine weiteren Argumente, stattdessen sollten die Möglichkeiten der Textanzeige selbst detaillierter beschrieben werden. Die einzelnen Befehle manipulieren die Textanzeige und werden durch einzelne Tasten aufgerufen.

MORE-Befehle:

- .<Text>** Sucht nach dem angegebenen <Text> in der Datei, wobei auf Groß- und Kleinschreibung nicht geachtet wird.
- /<Text>** Sucht nach dem angegebenen <Text> in der Datei, wobei auf Groß- und Kleinschreibung geachtet wird.
- B, Backspace** Springt eine Seite zurück, dieser Befehl ist jedoch nicht verfügbar, wenn die Eingabe aus dem logischen Gerät PIPE: erfolgt.
- Ctrl+L** Erneuert das Fenster (wenn es durch irgendwelche anderen Einflüsse verändert worden sein sollte).
- H, Help** Zeigt eine kurze aber sehr hilfreiche Liste aller verfügbaren Befehle an (aber leider in englischer Sprache). Wer aber ausreichend Englisch kann (es muß wirklich nicht allzuviel sein), der kann sich hier seinen gewünschten Befehl suchen.
- n** Wurde bei der vorherigen Suche nach einer Zeichenkette eine passende gefunden, so wird mit diesem Befehl das nächste Auftreten dieser Zeichenkette gesucht.
- Q, Ctrl+C** Beendet das Programm MORE, ein eventuell geöffnetes Fenster wird geschlossen, ein eigener Prozess beendet.
- Return** Schiebt den angezeigten Text um eine Zeile nach oben.
- Leertaste** Zeigt die nächste Seite des Textes an, wurde bereits die letzte Seite gezeigt, so wird MORE beendet.
- %N** Springt an die Stelle, an der n Prozent der Datei angezeigt wurden.
- <** Springt in die allererste Zeile der Datei. Dies ist bei einer Eingabe vom logischen Gerät PIPE: aus nicht möglich.
- >** Springt auf die letzte Zeile der Datei.

E Wurde in der Umgebungsvariable ENV:Editor ein ASCII-Editor eingestellt (beispielsweise MEMACS), so wird dieser gestartet und die angezeigte Datei zur Bearbeitung geladen.

Siehe auch:

ED, EDIT, MEMACS, MultiView

PAG Sandini

MOUNT

Syntax:

2.0/2.1: MOUNT [DEVICE] <device>: [FROM <file>]

3.0: MOUNT [DEVICE] <devicemuster>: [FROM <file>]

Schablone:

2.0/2.1/3.0: DEVICE/A, FROM/K

Pfad:

C:

Funktion:

Mit MOUNT wird ein angeschlossenes physikalisches Gerät angemeldet.

Beschreibung:

Dieser Befehl fügt dem AmigaDOS neue verfügbare Geräte hinzu. Der Begriff hat aber nichts mit eingelegten Disketten zu tun (bei diesen steht auch in der Ausgabe des Befehls INFO das Wort "[mounted]"). Dieser Befehl wird nicht nur in der Startup-Sequence benutzt, oftmals wird er auch bei der normalen Arbeit in einer Shell notwendig.

Die meisten physikalischen Geräte (wie Diskettenlaufwerke oder Speichererweiterungen) melden sich automatisch an, sei es durch entsprechende Chips auf der Platine oder durch spezielle Dateien im Verzeichnis SYS:Expansion, die bei der Ausführung der Startup-Sequence automatisch aufgerufen werden. Es gibt aber Geräte, die nicht immer benötigt werden und somit unnötig Speicher belegen, wenn sie nicht gebraucht werden, würden sie bei jedem Neustart des Amigas automatisch ins System eingebunden. Beispiele dafür sind etwa "Geräte" wie SPEAK:, PIPE:, PC0: oder RAD:.

Jedes anzumeldende Gerät besitzt einen eigenen Eintrag in der Datei DEVS:MountList, die jedesmal durchsucht wird, wenn der Befehl MOUNT verwendet wird. Es erscheint eine Fehlermeldung, wenn ein Eintrag in der MountList oder ein darin genannter Handler nicht gefunden werden kann.

Gerade bei der Verwendung der resetfesten RAM-Disk RAD: wird der Befehl MOUNT RAD: häufig verwendet. Es wäre ja töricht, wenn dieses speicherintensive Gerät jedesmal automatisch eingerichtet wird, auch wenn sie nicht benötigt wird. (Anmerkung: die RAD: kann mit dem Befehl "REMRAD" entfernt werden).

Seit der Version 2.1 kann der Befehl MOUNT auch Namensmuster von Gerätenamen verarbeiten. Dadurch können mit einem einzigen Befehl auch mehrere Geräte gleichzeitig angemeldet werden. Die Startup-Sequence benutzt diese Eigenschaft, um alle Geräte, die in der Schublade "DEVS/DOSDrivers" vorhanden sind, mit nur einem Befehl verfügbar zu machen.

Argumente:

DEVICE/A Name des Gerätes, das angemeldet werden soll. Wird das Schlüsselwort FROM nicht verwendet, so muß dieses Gerät einen eigenen Eintrag in der Datei DEVS:MountList besitzen. Näheres zu dieser mekrwürdigen MountList erfahren sie im nächsten Kapitel. Seit der Version 2.1 können auch Namensmuster des anzumeldenden Gerätes übergeben werden.

FROM/K Wenn in der Datei DEVS:MountList kein Eintrag für das gewünschte Gerät vorhanden ist, so muß sich MOUNT diese Informationen aus einer anderen Datei beschaffen. Aus welcher Datei sich MOUNT die benötigten Informationen holen soll, kann hinter diesem Schlüsselwort angegeben werden. Befindet sich diese Datei nicht im aktuellen Verzeichnis, muß der korrekte Pfadname eingegeben werden.

Siehe auch:

Kapitel "Die MountList".

MOUSEBLANKER

Syntax:

2.0: Befehl nicht implementiert

2.1/3.0: MOUSEBLANKER [CX_PRIORITY <priority>]

Schablone:

2.0: Befehl nicht implementiert

2.1/3.0: CX_PRIORITY/K/N

Pfad:

Extras:Tools/Commodities

Funktion:

Startet das Programm, daß nach einer gewissen Zeit ohne Mausbewegung den Mauszeiger abschaltet.

Beschreibung:

Der MOUSEBLANKER ist ein kleines gewöhnungsbedürftiges aber auch sehr nützliches Commodity, das den Mauszeiger nach einer gewissen Zeit versteckt, wenn die Maus in diesem Zeitraum nicht mehr bewegt wurde. Der Zeiger erscheint sofort wieder, sobald die Maus eine Bewegung meldet.

Wie alle anderen Commodities, sollte auch MOUSEBLANKER von der Shell mit dem Befehl RUN gestartet werden.

Argument:

CX_PRIORITY/K Siehe Beschreibung des Befehles CLICKTOFRONT

Siehe auch:

AUTOPOINT, BLANKER, CLICKTOFRONT, EXCHANGE, FKEY, IHELP, NOCAPSLOCK.

MULTISCAN

Syntax:

2.1/3.0: MULTISCAN [HBSTRT=<n>] [HBSTOP=<n>] [HSSTRT=<n>]
[HSSTOP=<n>] [VBSTRT=<n>] [VBSTOP=<n>]
[VSSTRT=<n>] [VSSTOP=<n>] [MINROW=<n>]
[MINCOL=<n>] [TOTROWS=<n>] [TOTCLKS=<n>]
[BEAMCON0=<n>]

2.0: Befehl nicht implementiert

Schablone:

2.1/3.0: HBSTRT/K,HBSTOP/K,HSSTRT/K,HSSTOP/K,
VBSTRT/K,VBSTOP/K,VSSTRT/K,VSSTOP/K,MINROW/K,
MINCOL/K,TOTROWS/K,TOTCLKS/K,BEAMCON0/K

2.0: Befehl nicht implementiert

Pfad:

DEVS: Monitors

Funktion:

Bei diesem Befehl handelt es sich um einen weiteren Monitortreiber, der nur bei Verwendung eines Multiscan-Monitors benutzt werden sollte.

Beschreibung:

Die Beschreibung des Befehls ADDMONTIOR gilt für Multiscan sinngemäß, weshalb ich Sie auf den Befehl ADDMONITOR verweise - insbesondere beachten Sie bitte den Hinweis am Ende der Befehlsbeschreibung von ADDMONITOR.

Siehe auch:

ADDMONITOR

MULTIVIEW

Syntax:

2.0/2.1: Befehl nicht implementiert.

3.0: MULTIVIEW [[FILE] <filename>] [CLIPBOARD] [CLIPUNIT <nn>]
[SCREEN] [PUBSCREEN <screen>] [REQUESTER] [BOOKMARK]
[FONTNAME] [FONTSIZE <nn>] [BACKDROP] [WINDOW]

Schablone:

2.0/2.1: Befehl nicht implementiert.

3.0: FILE,CLIPBOARD/S,CLIPUNIT/K/N,SCREEN/S,PUBSCREEN/K,
REQUESTER/S,BOOKMARK/S,FONTNAME/K,FONTSIZE/K/N,
BACKDROP/S,WINDOW/S

Pfad:

SYS: Utilities

PAG Sandini

Funktion:

Startet das gleichnamige Universalanzeigeprogramm.

Beschreibung:

Zunächst mag die Fülle der möglichen Argumente abschreckende Wirkung haben, aber der größte Teil ist leicht verständlich und sehr nützlich bzw. wichtig, um allen Aufgaben gerecht zu werden. Schließlich kann MultiView wesentlich mehr als sein Vorgänger MORE. MultiView kann jegliche Art von Information von der Textdatei bis zum Lied und vom Bild bis zum Sample ausgeben. MultiView ist so intelligent, daß er unter Anleitung auch auf neue Arten von Daten angepaßt werden kann, diese Anleitung muß nur in der Schublade "DataTypes" vorhanden sein.

MultiView und Bilder

MultiView hat eine begrenzte Menge von graphischen Möglichkeiten, so kann beispielsweise das Programm die Farben eines Bildes an die Anzahl der darstellbaren Farben auf dem Bildschirm anpassen. Gerade bei Bildern ist es ratsam, mit dem Schlüsselwort SCREEN MultiView dazu zu bewegen, daß ein eigener Screen zur Darstellung der Bilder verwendet wird, um

Schwierigkeiten mit der Farbdarstellung zu vermeiden. In einigen Darstellungsmodi ist es sogar möglich, Bildausschnitte zu kennzeichnen und in das ClipBoard zu kopieren.

MultiView und Hypertext

Comodore hat MultiView einen Übersetzer für Hypertext-Dateien verpaßt. Hypertext ist ein System, daß es ermöglicht, auch vom Programm aus beliebig detaillierte und komplexe Hilfestellungen zu bestimmten Programmen und Programmpunkten in fast jeder Situation zu leisten. Derzeit ist noch gar nicht absehbar, welche vielseitigen Möglichkeiten darin verborgen sind, die Entwicklung wird es zeigen.

Argumente:

FILE	Hier sollte der Name der Datei angegeben werden, deren Daten ausgegeben werden sollen. Es sich dabei um eine Text-, Musik-, Sample- oder Bilddatei handeln. Wird kein Dateiname angegeben, so ermöglicht ein Filerequester eine sehr unkomplizierte Dateiauswahl.
CLIPBOARD/S	Dieser Schalter überstimmt eine eventuell angegeben Quell-Datei, MultiView holt die auszugebenden Daten vom Gerät CLIPS:. Ein kleiner Hinweis: Im Gegensatz zu CONCLIP, das einen Text direkt im Gerät und nicht in einer Datei ablegt, erstellt MultiView in CLIPS: eine Datei namens "0".
CLIPUNIT/K/N	Hier kann die Nummer der Clipboard-Einheit angegeben werden, die benutzt werden soll. Erlaubt sind Werte zwischen 0 und 255. Diese Angabe ist jedoch normalerweise nicht nötig, es sei denn, sie wollen gezielt Daten retten bzw. löschen. Wird kein besonderer Wert übergeben, so verwendet MultiView die Einheit 0.
SCREEN/S	Wird dieser Schalter gesetzt, öffnet MultiView einen eigenen Screen, dessen Anzahl der darstellbaren Farben sich an der darzustellenden Datei orientiert.
PUBSCREEN/K	Hiermit kann das MultiView-Fenster in einem beliebi-

gen Public Screen geöffnet werden. In welchen, sagt der nachfolgende Name. Derzeit ist lediglich der Workbench-Screen ein richtiger Public Screen, Bildschirme von anderen Programmen dulden in der Regel keine Eindringlinge.

REQUESTER/S

Mit diesem Schlüsselwort wird MultiView dazu überredet, sämtliche Meldungen über Systemrequester kund zu tun und das Shell-Fenster, welches normalerweise diese Meldungen erhält, in Frieden zu lassen.

BOOKMARK/S

Hier benutzt MultiView eine sogenannte "Bookmark"(Lesezeichen)- Datei, wenn eine derartige Datei existiert.

FONTNAME/K

Soll ein Text dargestellt werden, so kann hier die Schriftart bestimmt werden, in der der Text erscheinen soll. Im Gegensatz zur Shell kann MultiView auch Compugraphic- und Proportional-Schriften korrekt verarbeiten.

FONTSIZE/K/N

In Verbindung mit FONTNAME muß natürlich auch eine Schriftgröße vorgegeben werden. Ist eine angegebene Größe nicht verfügbar, so wird eine benachbarte Größe verwendet.

BACKDROP/S

Mit diesem Schalter wird MultiView dazu veranlaßt, das Textanzeige-Fenster hinter allen anderen zu öffnen. Dies sieht auf normalen Screens sehr professionell aus.

WINDOW/S

Mit diesem Argument wird MultiView dazu überredet, keinen Filerequester zu öffnen, falls keine Quelldatei angegeben wurde. Diese Möglichkeit ist dafür vorgesehen, MultiView von der Startup-Sequenz schon aufzurufen, so daß das Programm ständig im Hintergrund vorhanden ist.

Siehe auch:
MORE

NEWCLI

Syntax:

2.0/2.1/3.0:

NEWCLI [[WINDOW] <Fensterspezifikation>] [[FROM] <startup-file>]

Schablone:

2.0/2.1/3.0: WINDOW, FROM

Pfad:

in die Shell integriert

Funktion:

Startet einen neuen Shell-Prozess und öffnet ein Fenster dafür.

Beschreibung:

Mit diesem Befehl ist es möglich, vom AmigaDOS aus die Multitasking-Fähigkeiten des Amigas auszunutzen. In den früheren AmigaDOS-Versionen schuf der Befehl NEWCLI einen neuen unabhängigen CLI-Prozess (CLI steht für Command Line Interpreter), der neben einem eigenen Fenster auch einen eigenen Stapelspeicher (Stack) und eine eigene Priorität besaß. Diese Werte wurden beim Start vom startenden CLI übernommen.

Mit Einführung der Version 1.3 wurde der CLI durch eine verbesserte Shell ergänzt, die vielfältigere Möglichkeiten zur Bearbeitung der Befehlseingabe bot. Es wurde dazu ein neues Konsolen-Fenster Namens "NEWCON:" bereitgestellt, daß im wesentlichen für die verbesserten Eigenschaften verantwortlich war. Das alte Fenster "CON:" war trotzdem weiterhin verfügbar.

Seit der Version 2.0 wurde CON: durch seinen Nachfolger NEWCON: vollkommen abgelöst, NEWCON: nam den Namen seines Vorgängers an. Waren vor der Version CLI und Shell zwei sehr unterschiedliche textorientierte Benutzeroberflächen, so existieren zwar beide Begriffe noch nebeneinander, sie unterscheiden sich aber in keinsten Weise voneinander. Die Zeiten der Rassenunterschiede sind endgültig vorbei. Daher werden wir nur den Befehl NEWCLI ausführlich erklären, bei der nachfolgenden Beschreibung des Befehles NEWSHELL verweisen wird auf die von NEWCLI.

Argumente:

WINDOW Wird kein derartiges Argument übergeben, so wird ein Fenster in einer fest vorgegebenen Position und Größe geöffnet. Dieses Argument ermöglicht dem Anwender, das Fenster in Position, Größe, Titel und verschiedenen anderen Eigenschaften seinen ganz persönlichen Bedürfnissen anzupassen. Dabei ist jedoch darauf zu achten, daß die Grenzen, die durch den Bildschirmmodus vorgegeben sind, nicht überschritten werden, sonst versagt der Befehl seinen Dienst. Es ist auch möglich andere interaktive AmigaDOS-Geräte als Konsole zu verwenden, beispielsweise AUX:. Hier wäre es denkbar, daß der Amiga von einem anderen Computer (am besten einem weiteren Amiga) aus bedient wird. Die genaue Struktur der Fensterspezifikation wurde zu Beginn dieses Kapitels genau beschrieben.

FROM Wird ein neuer Shell-Prozess gestartet, so versucht NEWCLI noch vor der Benutzer die Kontrolle erhält, die Befehlsdatei "S:Shell-Startup" auszuführen, sofern diese existiert. Wenn jedoch an deren Stelle eine andere Befehlsdatei ausgeführt werden soll, so kann die entsprechende Datei hinter dem Schlüsselwort FROM genannt werden. Enthält diese Script-Datei Befehle, die Fehler melden oder gar falsch sind, so wird die Ausführung der Befehlsdatei abgebrochen, und dem Anwender die Kontrolle über dem Prozess ohne die Bearbeitung der späteren Befehle übergeben. Eine entsprechende Fehlermeldung vor der ersten Eingabeaufforderung weist aber darauf hin, daß irgendetwas schief gelaufen ist.

Sollte die gewünschte Datei nicht im aktuellen Verzeichnis stehen, so ist der korrekte Pfad einzugeben, damit AmigaDOS die Datei findet und ausführen kann.

Beispiele:

Der folgende Befehl stammt aus meiner Datei "S:User-Startup" und bewirkt, daß im Hintergrund ein Shell-Fenster mit den Maßen 680 x 510 Bildpunkte geöffnet wird und kein Zieh- und Größen-Gadget enthält. Der Titel ist hier eigentlich überflüssig, da ein Backdrop-Fenster keine Titelleiste besitzt.

Wenn das Fenster geöffnet ist, so wird die Datei "S:User-Startup1" ausgeführt.

newcli "con:0/0/680/510/ A G M /backdrop nodrag nosize" s:User-startup1

Bemerkungen:

Wird ein neuer Shell-Prozess gestartet, so übernimmt er vom startenden Prozess folgende Einstellungen: Größe des Stapelspeichers, das aktuelle Verzeichnis, durch den Befehl Alias definierte Abkürzungen, die Priorität des Prozesses und den Suchpfad für die Befehle.

Welche Möglichkeiten zur Bearbeitung der Befehlszeilen vorhanden sind und welche verschiedene Fenster-Eigenschaften definiert werden können, wurde bereit zu Beginn dieses Kapitels erklärt.

Siehe auch:

NEWSHELL

PAG Sandini

NEWSHELL

Syntax:

2.0/2.1/3.0: NEWSHELL [[WINDOW] <Fensterspezifikation>]
[[FROM] <startup-file>]

Schablone:

2.0/2.1/3.0: WINDOW, FROM

Pfad:

in die Shell integriert

Funktion:

Startet einen neuen Shell-Prozess und öffnet ein Fenster dafür.

Beschreibung:

Dieser Befehl gleicht bis ins letzte Detail dem Befehl NEWCLI, so daß ich auf die Beschreibung dort verweisen darf.

Siehe auch:

NEWCLI

NOCAPSLOCK

Syntax:

2.0/2.1/3.0: NOCAPSLOCK [CX_PRIORITY <n>]

Schablone:

2.0/2.1/3.0: CX_PRIORITY/K/N

Pfad:

Extras:Tools/Commodities

Funktion:

Schaltet die Taste "Caps Lock" ab.

Beschreibung:

Dieser Befehl machte die Taste "Caps Lock" (nur Großbuchstaben) unwirksam. Der tiefere Sinn gibt sich vermutlich nur denen Preis, die übermäßig oft die falsche Taste erwischen. Hiermit kann zumindest die Gefahr eingedämmt werden, daß anstatt dem "a" die "Caps Lock"-Taste erwischt wird, so daß der Rest des Wortes groß geschrieben wird. LLERDINGS ist darauf hinzuweisen, daß gar kein Zeichen erscheint, wenn die Taste inaktiviert ist und man versehentlich darauf drückt.

Wie auch alle anderen Commodities, ist auch NOCAPSLOCK nicht in der Lage, einen eigenständigen Prozess zu starten, so daß die Shell lahm gelegt wird, solange der Befehl läuft, es sei denn, NOCAPSLOCK wurde mit RUN gestartet. Ist das Programm bereits aktiv, so kann es durch einen weiteren Aufruf von NOCAPSLOCK wieder aus dem System entfernt werden. In diesem Fall bricht NOCAPSLOCK sofort ab, so daß beim zweiten Aufruf kein RUN eingegeben werden braucht.

Argumente:

CX_PRIORITY/K/N Siehe die Beschreibung des Befehls CLICK TOFRONT.

Siehe auch:

AUTOPOINT, BLANKER, CLICKTORFRONT, EXCHANGE, FKEY, IHELP, MOUSE-BLANKER.

NOFASTMEM

Syntax:

2.0/2.1/3.0: NOFASTMEM

Schablone:

2.0/2.1/3.0: -

Pfad:

SYS:System

Funktion:

Sperrt das eventuell vorhandene FAST-RAM.

Beschreibung:

Dieser Befehl reserviert sich selbst den kompletten verfügbaren FAST-RAM-Speicher, so daß er von anderen Programmen nicht mehr genutzt werden kann. Wer jetzt den Kopf schüttelt, weil er selbst jedesmal händeringend die einzeln verstreuten Bytes im RAM zusammenkratzt, um das Programm doch noch zum Laufen zu bewegen, der sollte doch einmal einen Blick auf den geschichtlichen Hintergrund dieses Befehls werfen.

Der Befehl lähmt die Shell, wenn er nicht durch den Befehl RUN einen eigenen Prozess erhält. Abgebrochen werden kann er durch die Tastenkombination Ctrl-C oder dem entsprechenden BREAK-Befehl. Wird NOFASTMEM zum zweiten Mal aufgerufen, so wird der Befehl ebenfalls inaktiviert.

Beispiel:

Der erste Aufruf mit dem Befehl RUN weist NOFASTMEM einen eigenen Task zu, durch den zweiten Aufruf wird der Prozess entfernt:

```
1.Work:DOS3.0> run nofastmem
```

```
[CLI 3]
```

```
1.Work:DOS3.0> status
```

```
Process 1: Loaded as command: status
```

Process 2: Loaded as command: C:ConClip

Process 3: Loaded as command: nofastmem

1.Work:DOS3.0> nofastmem

1.Work:DOS3.0> status

Process 1: Loaded as command: status

Process 2: Loaded as command: C:ConClip

Bemerkungen:

Die im Amiga eingebauten Custom-Chips (das sind Coprozessoren, sozusagen Sklaven des Hauptprozessors) können nur auf einen bestimmten Teil des Speichers zugreifen, steht einem Amiga (in der Regel durch Einbau von Speichererweiterung) mehr RAM zur Verfügung, können die Custom-Chips auf darin abgelegte Daten nicht zurückgreifen, dieser Teil ist einzig dem Hauptprozessor vorbehalten. Der Bereich, der sich der Prozessor mit den Custom-Chips teilen muß, wird CHIP-RAM genannt, der darüber hinaus reichende Speicher ist das sogenannte FAST-RAM.

Bei den ersten Amigas konnten die Custom-Chips nur 512 KByte CHIP-RAM verwenden, sie wurden aber im Laufe der Zeit schrittweise verbessert, so daß über eine Zwischenstufe von 1 MB mittlerweile 2 MB CHIP-RAM vorhanden sind.

Nun kam es vor allem bei der älteren Software vor, daß die Programmierer nicht mit dem Vorhandensein eines FAST-RAM rechneten und einfach alle Daten und darunter eben auch die für die Custom-Chips sorglos im Speicher ablegten. Als dann aber die ersten Speichererweiterungen auftauchten, traten große Probleme auf, weil die Custom-Chips auf einmal bestimmte Daten nicht mehr erreichen konnten. Nicht selten war ein Rechnerabsturz die Konsequenz. Im besten Fall kamen undefinierte Laute aus dem Lautsprecher oder war ein futuristisch-entartetes Gemälde zu sehen, für das wir wohl noch kein Kunstverständnis entwickelt haben. In schwerwiegenderen Fällen wurden undefinierte Zeichen auf die Diskette geschrieben, bisweilen wurde sie dadurch aber auch zersört.

Durch den Befehl NOFASTMEM, der den zusätzlichen Speicher für sich belegt und dadurch sperrt, konnten diese Probleme beseitigt werden.

NTSC

Syntax:

2.1/3.0: NTSC [HBSTRT=<n>] [HBSTOP=<n>] [HSSTRT=<n>]
[HSSTOP=<n>] [VBSTRT=<n>] [VBSTOP=<n>]
[VSSTRT=<n>] [VSSTOP=<n>] [MINROW=<n>]
[MINCOL=<n>] [TOTROWS=<n>] [TOTCLKS=<n>]
[BEAMCON0=<n>]

2.0: Befehl nicht implementiert

Schablone:

2.1/3.0: HBSTRT/K,HBSTOP/K,HSSTRT/K,HSSTOP/K,
VBSTRT/K,VBSTOP/K,VSSTRT/K,VSSTOP/K,MINROW/K,
MINCOL/K,TOTROWS/K,TOTCLKS/K,BEAMCON0/K

2.0: Befehl nicht implementiert

Pfad:

DEVS:Monitors

Funktion:

Bei diesem Befehl handelt es sich um einen weiteren Monitortreiber, der nur bei Verwendung eines NTSC-Monitors benutzt werden sollte.

Beschreibung:

Die Beschreibung des Befehls ADDMONTIOR gilt für NTSC sinngemäß, weshalb ich Sie auf den Befehl ADDMONITOR verweise - insbesondere beachten Sie bitte den Hinweis am Ende der Befehlsbeschreibung von ADDMONITOR.

Siehe auch:

ADDMONITOR

OVERSCAN

Syntax:

2.0: OVERSCAN [[FROM] <file>] [EDIT] [USE] [SAVE]

2.1/3.0: OVERSCAN [[FROM] <file>] [EDIT] [USE] [SAVE]
[PUBSCREEN <screen>]

Schablone:

2.0: FROM,EDIT/S,USE/S,SAVE/S

2.1/3.0: FROM,EDIT/S,USE/S,SAVE/S,PUBSCREEN/K

Pfad:

SYS:Prefs

Funktion:

Es kann die Bildschirmjustierung angezeigt und verändert werden.

Beschreibung:

Wird OVERSCAN ohne ein weiteres Argument oder mit dem Schlüsselwort "EDIT" aufgerufen, wird das Programm der Prefs-Schublade gestartet. Wird ein Dateiname angegeben und besitzt diese Datei Informationen zu den Voreinstellungen (sie wurde im OVERSCAN-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert), können diese schon vorher definierten Einstellungen übernommen werden.

Argumente:

FROM Name einer Datei, die im OVERSCAN-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert wurde. Was mit diesen bereits definierten Einstellungen passiert, wird mit den anschließenden Schlüsselwörtern festgelegt.

EDIT/S Startet das Programm "OVERSCAN" der Prefs-Schublade, wie wenn das Icon auf der Workbench in der Schublade Prefs angeklickt worden wäre. Dies ist auch dann der Fall, wenn überhaupt kein Argument übergeben wird.

- USE/S** Wurde ein Dateiname einer OVERSCAN-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen aktiviert. Beim Neustart des Rechners gelten jedoch wieder die alten Einstellungen.
- SAVE/S** Wurde ein Dateiname einer OVERSCAN-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen gespeichert und damit bei jedem Neustart des Rechners aktiviert.
- PUBSCREEN/K** Wird hier ein Name eines anderen Screens eingegeben, so wird das OVERSCAN-Fenster auf diesen Screen geöffnet.

PAG Sandini

PAL

Syntax:

2.1/3.0: PAL [HBSTRT=<n>] [HBSTOP=<n>] [HSSTRT=<n>]
[HSSTOP=<n>] [VBSTRT=<n>] [VBSTOP=<n>]
[VSSTRT=<n>] [VSSTOP=<n>] [MINROW=<n>]
[MINCOL=<n>] [TOTROWS=<n>] [TOTCLKS=<n>]
[BEAMCON0=<n>]

2.0: Befehl nicht implementiert

Schablone:

2.1/3.0: HBSTRT/K,HBSTOP/K,HSSTRT/K,HSSTOP/K,
VBSTRT/K,VBSTOP/K,VSSTRT/K,VSSTOP/K,MINROW/K,
MINCOL/K,TOTROWS/K,TOTCLKS/K,BEAMCON0/K

2.0: Befehl nicht implementiert

Pfad:

DEVS:Monitors

Funktion:

Bei diesem Befehl handelt es sich um einen weiteren Monitortreiber, der nur bei Verwendung eines PAL-Monitors benutzt werden sollte.

Beschreibung:

Die Beschreibung des Befehls ADDMONTIOR gilt für PAL sinngemäß, weshalb ich Sie auf den Befehl ADDMONITOR verweise - insbesondere beachten Sie bitte den Hinweis am Ende der Befehlsbeschreibung von ADDMONITOR.

Siehe auch:

ADDMONITOR

PALETTE

Syntax:

2.0/2.1/3.0: PALETTE [[FROM] <file>] [EDIT] [USE] [SAVE]

Schablone:

2.0/2.1/3.0: FROM,EDIT/S,USE/S,SAVE/S

Pfad:

SYS:Prefs

Funktion:

Es können die Farbeinstellungen angezeigt und verändert werden.

Beschreibung: PAG Sandini

Wird PALETTE ohne ein weiteres Argument oder mit dem Schlüsselwort "EDIT" aufgerufen, wird das Programm der Prefs-Schublade gestartet.

Wird ein Dateiname angegeben und besitzt diese Datei Informationen zu den Voreinstellungen (sie wurde im PALETTE-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert), können diese schon vorher definierten Einstellungen übernommen werden.

Argumente:

FROM Name einer Datei, die im PALETTE-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert wurde. Was mit diesen bereits definierten Einstellungen passiert, wird mit den anschließenden Schlüsselwörtern festgelegt.

EDIT/S Startet das Programm "PALETTE" der Prefs-Schublade, wie wenn das Icon auf der Workbench in der Schublade Prefs angeklickt worden wäre. Dies ist auch dann der Fall, wenn überhaupt kein Argument übergeben wird.

- USE/S Wurde ein Dateiname einer PALETTE-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen aktiviert. Beim Neustart des Rechners gelten jedoch wieder die alten Einstellungen.
- SAVE/S Wurde ein Dateiname einer PALETTE-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen gespeichert und damit bei jedem Neustart des Rechners aktiviert.

PAG Sandini

PARK

Syntax:

2.0/2.1/3.0: PARK [{<partition>}] [INHIBIT]

Schablone:

2.0/2.1/3.0: <partition>/M,INHIBIT/S

Pfad:

Sys:System (nur bei Systemen mit Festplatte)

Funktion:

Führt die Festplatte in eine Schutzstellung.

Beschreibung:

Daß Festplatten gegenüber äußeren mechanischen Einflüssen sehr empfindlich sind, wird schon allein dadurch verständlich, wenn man bedenkt, auf welch engen Raum ein **Unmenge** von Daten (Bis zu mehreren hundert Megabyte auf einer 2.5"-Festplatte) untergebracht sind. Selbst eine leichte Erschütterung kann eine Festplatte zerstören - oder zumindest die darauf gespeicherten Daten. Wird jedoch die Oberfläche einer Festplatte physikalisch beschädigt, etwa weil der sonst darüber schwebende Schreib-/Lesekopf sie berührt hat, so kann erforderliche Reparatur ganz schön teuer werden. Ein weiterer empfindlicher Punkt ist das Ein- und Ausschalten der Festplatte - auch hier sind die Geräte sehr empfindlich. Hier greift nun der Befehl PARK an. Er wird dazu verwendet, der Festplatte mitzuteilen, daß demnächst der Computer ausgeschaltet wird, und sie deswegen in "Deckung" gehen soll. Normalerweise wird dann der Schreib-/Lesekopf in eine besonders geschützte Stellung gefahren, so daß er zumindest vor leichten und mittelschweren Erschütterungen in Sicherheit ist. Erfolgt aber nach dem Befehl PARK wieder ein Zugriff auf die Festplatte, so wird der Kopf wieder aus der geschützten Stellung herausgefahren. Sie sollten daher immer absolut sicher sein, daß dieser Befehl die allerletzt ausgeführte Aktion des Amigas ist, sonst hilft alles nichts.

Argumente:

<partition>/M Ein oder mehrere Namen von Partitionen, die durch den Befehl PARK stillgelegt werden. Mehrere Partitionen sollten

jedoch nur dann genannt werden, wenn die Option INHIBIT verwendet wird, andernfalls sollte nur die Partition eingegeben werden, die über eine PARK-Spur verfügt. Wird überhaupt kein Gerät übergeben, so ist "DH0:" oder "HD0:" voreingestellt.

INHIBIT/S Mit INHIBIT wird der Zugriff auf die angegebene Partition vollkommen gesperrt, es kann dann weder gelesen noch geschrieben werden, da es im Endeffekt die Partition aus dem System entfernt. Dies ist besonders dann sinnvoll, wenn persönliche Daten besonders geschützt werden sollen. Es muß jedoch erwähnt werden, daß in diesem Fall die Festplatte nicht geparkt im eigentlichen Sinn wird, vielmehr wird softwareseitig die Festplatte aus dem System gestrichen. Die Sperre kann durch ein Reset - und nur dadurch - wieder aufgehoben werden.

Es ist zwar möglich, auch die Boot-Partition auf diese Weise zu sperren, doch kann ich nur ausdrücklich davor warnen. Es ist dann nämlich nicht mehr möglich, irgend einen Teil des dort vorhandenen Betriebssystems zu verwenden, so daß sehr wahrscheinlich der gesamte Amiga stillgelegt wird, ein Arbeiten unmöglich wird.

Bemerkungen:

Dieser Befehl ist nur bei Amiga-Systemen vorhanden, die über mindestens eine Festplatte verfügen. Da der Schalter INHIBIT einen tiefen Eingriff in die Systemstrukturen verursacht, sollte von einer Verwendung abgesehen werden. Verwenden Sie zum Schutz der Daten lieber den Befehl LOCK.

Es gibt jetzt schon viele Festplatten, die nach dem Abschalten des Rechners automatisch in Park-Position fahren (die Ausnutzung der Rotation der Festplatte zur Erzeugung von Strom macht's möglich). Es muß also auch bei Festplatten-Systemen nicht unbedingt dieser Befehl vorhanden sein. Wenn Sie aber irgenwo auf den Befehl PARK stoßen, so sollten Sie sich angewöhnen, ihn jedesmal unmittelbar vor dem Abschalten des Amigas aufzurufen, nach dem Befehl darf kein Zugriff auf die Festplatte mehr erfolgen, sonst ist seine Wirkung wieder verpufft.

Siehe auch:

LOCK

PATH

Syntax:

2.0: PATH [{<dir>}] [ADD] [SHOW] [RESET] [QUIET] [REMOVE]

2.1/3.0: PATH [{<dir>}] [ADD] [SHOW] [RESET] [REMOVE] [QUIET]

Schablone:

2.0: PATH/M, ADD/S, SHOW/S, RESET/S, QUIET/S, REMOVE/S

2.1/3.0: PATH/M, ADD/S, SHOW/S, RESET/S, REMOVE/S, QUIET/S

Pfad:

in die Shell integriert

Funktion:

Der Suchpfad für Befehle wird geändert oder angezeigt.

Beschreibung:

Mit Hilfe dieses Befehls kann der Anwender AmigaDOS mitteilen, in welchen Verzeichnissen es die Befehle suchen soll, die er ständig eingibt. Wird PATH kein weiteres Argument übergeben, so werden die Verzeichnisse des gegenwärtigen Suchpfades in der gültigen Reihenfolge aufgelistet. Ganz zu Beginn, bevor noch keine weiteren Verzeichnisse hinzugefügt wurden, besteht der Suchpfad lediglich aus dem aktuellen Verzeichnis und dem Verzeichnis des logischen Gerätes C:, welches gewöhnlich das Verzeichnis "C" der Bootdiskette oder -festplatte ist.

Unter diesen Bedingungen ist es nicht möglich, solche Befehle wie etwa DPAT oder NOFASTMEM anzusprechen, da der Amiga diese nie finden würde, wenn nicht der komplette Pfadname angegeben wäre. Durch den Suchpfad ist es jedoch möglich, auch Befehle in anderen Verzeichnissen direkt zu verwenden, ohne ständig den lästigen Pfad komplett angeben zu müssen. Wird ein Befehl eingegeben, so durchsucht AmigaDOS der Reihe nach sämtliche im Suchpfad eingetragenen Verzeichnisse nach einem Befehl. Wird ein derartiger Befehl gefunden, so wird die Suche abgebrochen und das

Programm gestartet. So werden automatisch Schwierigkeiten vermieden, wenn gleichzeitig mehrere Programme in verschiedenen Verzeichnissen den eingegeben Namen besitzten. Erst wenn der Befehl auch im letzten Verzeichnis nicht fündig wird, erscheint die Fehlermeldung "unknown command".

Bei meinem System sind beispielsweise folgende Verzeichnisse im Suchpfad eingetragen:

```
1.Work:DOS3.0> path
Current_directory
Ram Disk:
Workbench:C
Workbench:Utilities
Workbench:Rexxc
Workbench:System
Workbench:S
Workbench:Prefs
Workbench:WBStartup
Workbench:Tools
Workbench:Tools/Commodities
Work:TeX/PasTeX/bin
Work:TeX/PasTeX/bigbin
Work:TeX/METAFONT/bin20
Work:DH0/c
C:
```

Wenn Sie die obige Liste genauer betrachten, so fällt vielleicht die Zeile Work:DH0/c ins Auge. Das Verzeichnis DH0: stammt von meinem ersten Amiga und ist der komplette Inhalt dessen Festplatte, darin sind im Verzeichnis "C" natürlich wiederum Befehle enthalten, die möglicherweise im neuen Betriebssystem nicht (mehr) vorhanden sind.

AmigaDOS durchsucht die Verzeichnisse von oben nach unten in der angegebenen Reihenfolge. Gebe ich beispielsweise einen Befehl ein, der im neuen Betriebssystem vorhanden ist (und gegenüber dem alten verbessert wurde), so wird der neue Befehl ausgeführt, da das Verzeichnis "Workbench:C" in der Reihenfolge auf dem 3. Platz steht. Ist aber ein anderer Befehl nicht mehr vorhanden, so wird AmigaDOS erst im Verzeichnis "Work:DH0/C" fündig.

(Anmerkung: Ich unterlaufe die Philosophie des Amigas hier zwar, weil eigentlich das Verzeichnis C: erst am Ende der Liste vorgesehen ist, aber zur Erklärung halte ich dieses Beispiel für sehr geeignet.)

Beachten Sie, daß das aktuelle Verzeichnis immer ganz oben im Suchpfad steht, also erst in diesem nach dem eingegebenen Befehl gesucht wird. Gerade für Programmierer ist dieser Sachverhalt sehr nützlich, da er eine lauffähige Vorversion in einem Verzeichnis des Suchpfades ablegen kann, während er weitere Verbesserungen vornehmen wird. Will er während seiner Arbeit einmal testen, ob die Verbesserungen auch korrekt und seinen Vorstellungen entsprechend ablaufen, kann er sicher sein, daß das zu testende Programm gestartet wird, wenn es im aktuellen Verzeichnis vorhanden ist. Werden mit PATH weitere Verzeichnisse in den Suchpfad aufgenommen, so werden sie in der Reihenfolge zwischen den bereits vorhandenen Verzeichnissen und dem letzten Eintrag "C:" eingefügt. Die Philosophie des Amigas will jedem anderen eine Chance geben, bevor die eigenen Programme gestartet werden.

Wird ein Verzeichnis einer Diskette in den Suchpfad eingebunden und die Diskette aus dem System entfernt, so fordert AmigaDOS den Benutzer jedoch nicht auf, die Diskette einzulegen, damit dort nach dem Befehl gesucht werden kann. Wäre dieser Befehl auf der Diskette vorhanden gewesen, so schlägt die Suche fehl, es erscheint der Fehler "unknown command". Andererseits wird aber verlangt, die Diskette einzulegen, wenn das Verzeichnis in den Suchpfad aufgenommen werden soll. AmigaDOS ist doch sehr mißtrauisch, und untersucht zuerst, ob das Verzeichnis wirklich existiert.

Es ist nicht möglich, ein Verzeichnis eines mit den Schlüsselwörtern DEFER oder PATH des Befehls ASSIGN erklären logischen Gerätes in den Suchpfad aufzunehmen, wenn keine geeignete Diskette in einem Laufwerk einliegt.

Argumente:

PATH/M

Hier werden die vollständigen Pfade der in den Suchpfad aufzunehmenden Verzeichnisse eingegeben. Die Anzahl der einzugebenden Pfadnamen ist lediglich durch die Zeilenlänge des Konsolen-Fensters begrenzt.

ADD/S

Mit diesem Schlüsselwort wird PATH mitgeteilt, das der Suchpfad durch das angegebene Verzeichnis ergänzt wer-

den soll. Dies ist jedoch voreingestellt, wenn außer Pfadnamen keine weiteren Schlüsselwörter angegeben werden.

SHOW/S

Dieses Schlüsselwort veranlaßt PATH, die aktuelle Liste der Verzeichnisse, die nach Befehlen durchsucht werden, auszugeben. Auch dieses Schlüsselwort wird sehr selten benutzt, da die Eingabe des Befehls PATH ohne jegliches Argumente das gleiche bewirkt.

RESET/S

Dieser Schalter sollte entweder als einziges Argument von PATH oder zumindest als erstes noch vor irgendwelchen anderen verwendet werden. RESET löscht den gesamten aktuellen Suchpfad bis auf die voreingestellten (aktuelles Verzeichnis und "C:"). Diese Änderung ist jedoch immer nur für einen Prozess gültig.

QUIET/S

Wird einmal ein Verzeichnis in den Suchpfad aufgenommen, so versucht PATH jedesmal, wenn die Liste ausgegeben werden soll, alle gesetzten Directories auch zu finden. So kann der Benutzer aufgefordert werden, Disketten einzulegen, wenn darauf ebenfalls Verzeichnisse im aktuellen Suchpfad vorhanden sind. Um diese Requester zu vermeiden, kann das Schlüsselwort QUIET verwendet werden. Es erscheint dann lediglich der Name der Diskette in der ausgegebenen Liste.

REMOVE/S

Ebenso wie neue Verzeichnisse in die Liste aufgenommen werden können, besteht auch die Möglichkeit, einzelnen Einträge wieder zu entfernen. Zu diesem Zweck ist das Schlüsselwort REMOVE entstanden, daß die angegebenen Verzeichnisse aus dem Suchpfad entfernt, alle anderen jedoch unverändert läßt.

Bemerkungen:

Wird ein neuer Prozess gestartet, so übernimmt dieser den Suchpfad des Mutterprozesses. Daher hat es den Anschein, als ob die Suchpfade global für alle laufenden Prozesse gültig wären. Dem ist aber nicht so, in Wirklichkeit wirkt sich jede Änderung des Suchpfades nur auf den Prozess aus, in dem die

Änderung vorgenommen wird, alle anderen Prozesse erfahren davon nichts.

Es kann vorkommen, daß Sie ein leeres Verzeichnis löschen wollen, doch AmigaDOS verweigert es mit der Fehlermeldung "object in use". Dies kann dreierlei Gründe haben. Entweder ist dem Verzeichnis ein logisches Gerät mit dem Befehl ASSIGN zugewiesen worden, oder es befindet sich im Suchpfad eines Prozesses, oder es ist selbst ein aktuelles Verzeichnis eines anderen Prozesses. Hierbei spielt es jedoch keine Rolle, ob das Verzeichnis auch im Suchpfad des Prozesses ist, der AmigaDOS den Lösch-Auftrag erteilt. Es reicht, wenn irgendein Prozess das Verzeichnis im Suchpfad enthält. Ein Beispiel dazu:

```
5.Work:DOS3.0> makedir ram:test
```

```
5.Work:DOS3.0> path ram:test add
```

```
5.Work:DOS3.0> path
```

```
Current_directory
```

```
Ram Disk:
```

```
Workbench:Utilities
```

```
Workbench:Rexxc
```

```
Workbench:System
```

```
Workbench:S
```

```
Workbench:Prefs
```

```
Workbench:WBStartup
```

```
Workbench:Tools
```

```
Workbench:Tools/Commodities
```

```
Ram Disk:test
```

```
C:
```

PAG Sandini

```
4.Work:DOS3.0> delete ram:test
```

```
ram:test Not Deleted: object is in use
```

Das Verzeichnis "ram:Test" konnte im Prozess 4 nicht gelöscht werden, weil es im Suchpfad des Prozesses 5 enthalten ist.

Siehe auch:

ASSIGN, CD

PCD

Syntax:

2.0/2.1/3.0: PCD [[PAT] <newdir | muster>]

Schablone:

2.0/2.1/3.0: PAT

Pfad:

S:

Funktion:

PCD erweitert den Befehl CD, das jeweils vorhergehende aktuelle Verzeichnis wird gespeichert und kann wieder als solches deklariert werden.

Beschreibung:

Wie die Befehle DPAT und SPAT ist auch PCD ein sogenannter Makro-Befehl (auch Script- oder Befehlsdatei genannt). Er arbeitet wie der Befehl CD selbst, jedoch mit dem Unterschied, daß das aktuelle Verzeichnis gespeichert wird, bevor das neue als aktuelles Verzeichnis erklärt wird. So kann hinterher wieder das alte als aktuelles Verzeichnis deklariert werden, in dem PCD ohne Argumente eingegeben wird.

Dies verlangt jedoch, daß PCD vorher mindestens einmal mit einem Argument eingesetzt wurde, andernfalls wird kein Verzeichnis gespeichert, daß PCD ohne Argument als aktuelles deklarieren soll. Dies führt zu einem Fehler.

Argument:

PAT Name oder Namensmuster des Verzeichnisses, in das PCD springen soll. Gespeichert wird jedoch nicht das Zielverzeichnis, sondern das ursprüngliche Verzeichnis.

Bemerkung:

PCD ist auf die Multitasking-Fähigkeiten des Amigas vorbereitet, wird der Befehl in verschiedenen Prozessen verwendet, so springt jeder Prozess an

seine eigenen Marke wieder zurück.

Beispiele:

Der Befehl CD hat keinen Einfluß auf PCD:

```
5.Work:DOS3.0> pcd dh0:tex/metafont/metafont/pk/120
5.Work:DH0/TeX/MetaFont/metafont/pk/120> cd ram:Env
5.Ram Disk:ENV> cd hd1:TeX
5.Work:TeX> pcd
5.Work:DOS3.0>
```

Wird jedoch PCD wiederholt ohne Argument aufgerufen, so springt das aktuelle Verzeichnis ständig hin und her:

```
5.Work:DOS3.0> cd HD1:AmigaEinstieg/Bilder/Workbench
5.Work:AmigaEinstieg/Bilder/Workbench> pcd
5.Work:TeX> pcd
5.Work:AmigaEinstieg/Bilder/Workbench> pcd
5.Work:TeX>
```

PAG Sandini

Siehe auch:

CD, PROMPT

POINTER

Syntax:

2.0: POINTER [[FROM] <file>] [EDIT] [USE] [SAVE]

2.1/3.0: POINTER [[FROM] <file>] [EDIT] [USE] [SAVE]
[CLIPUNIT <n>] [NOREMAP]

Schablone:

2.0: FROM,EDIT/S,USE/S,SAVE/S

2.1/3.0: FROM,EDIT/S,USE/S,SAVE/S,CLIPUNIT/K/N,NOREMAP/S

Pfad:

SYS:Prefs

PAG Sandini

Funktion:

Der Mauszeiger kann verändert werden.

Beschreibung:

Wird POINTER ohne ein weiteres Argument oder mit dem Schlüsselwort "EDIT" aufgerufen, wird das Programm der Prefs-Schublade gestartet. Wird ein Dateiname angegeben und besitzt diese Datei Informationen zu den Voreinstellungen (sie wurde im POINTER-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert), können diese schon vorher definierten Einstellungen übernommen werden.

Argumente:

FROM Name einer Datei, die im POINTER-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert wurde. Was mit diesen bereits definierten Einstellungen passiert, wird mit den anschließenden Schlüsselwörtern festgelegt.

EDIT/S Startet das Programm "POINTER" der Prefs-

Schublade, wie wenn das Icon auf der Workbench in der Schublade Prefs angeklickt worden wäre. Dies ist auch dann der Fall, wenn überhaupt kein Argument übergeben wird.

USE/S

Wurde ein Dateiname einer POINTER-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen aktiviert. Beim Neustart des Rechners gelten jedoch wieder die alten Einstellungen.

SAVE/S

Wurde ein Dateiname einer POINTER-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen gespeichert und damit bei jedem Neustart des Rechners aktiviert.

CLIPUNIT/K/N

Hinter diesem optionalen Schlüsselwort muß die Nummer der Clipboard-Einheit eingegeben werden, die benutzt werden soll.

NOREMPAT/S

Dieser Schalter ermöglicht die Umwandlung der Farbreister der älteren Amigas in die Version 3.0. Dieser Schalter sollte jedoch nicht verwendet werden.

PREPCARD

Syntax:

- 2.0: Befehl nicht implementiert
- 2.1/3.0: PREPCARD [DISK] [RAM]

Schablone:

- 2.0: Befehl nicht implementiert
- 2.1/3.0: DISK/S, RAM/S

Pfad:

Extras:Tools

Funktion:

Dieser Befehl ist zwar bei allen Amigas mit Workbench Version 2.1 oder neuer vorhanden, kann aber nur eingesetzt werden, wenn der Amiga eine PCMCIA-Schnittstelle besitzt (Amiga 600 und Amiga 1200). Wird der Befehl bei anderen Amigas eingesetzt, erscheint die Fehlermeldung "No Card Slot".

Argumente:

- DISK/S Mit diesem Schlüsselwort wird eine eingelegte PCMCIA-Karte vorbereitet, als fiktives Diskettenlaufwerk zu arbeiten. Es können dann sämtliche Befehle die für eine Diskette gültig sind auch auf diese Karte angewandt werden, so beispielsweise auch FORMAT.
- RAM/S Mit diesem Schlüsselwort wird dem Amiga mitgeteilt, daß er die eingelegte PCMCIA-Karte als Speichererweiterung betrachten und entsprechend behandeln soll. Diese Karte wird dann in das vorhandene RAM eingebunden.

Siehe auch:

FORMAT, NOFASTMEM.

PRINTER

Syntax:

2.0: PRINTER [[FROM] <file>] [EDIT] [USE] [SAVE]

2.1/3.0: PRINTER [[FROM] <file>] [EDIT] [USE] [SAVE]
[PUBSCREEN <screen>] [UNIT]

Schablone:

2.0: FROM,EDIT/S,USE/S,SAVE/S

2.1/3.0: FROM,EDIT/S,USE/S,SAVE/S,PUBSCREEN/K,UNIT/S

Pfad:

SYS:Prefs

Funktion:

Es kann die Druckeranpassung angezeigt oder verändert werden.

Beschreibung:

Wird PRINTER ohne ein weiteres Argument oder mit dem Schlüsselwort "EDIT" aufgerufen, wird das Programm der Prefs-Schublade gestartet. Wird ein Dateiname angegeben und besitzt diese Datei Informationen zu den Voreinstellungen (sie wurde im PRINTER-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert), können diese schon vorher definierten Einstellungen übernommen werden.

Argumente:

FROM Name einer Datei, die im PRINTER-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert wurde. Was mit diesen bereits definierten Einstellungen passiert, wird mit den anschließenden Schlüsselwörtern festgelegt.

EDIT/S Startet das Programm "PRINTER" der Prefs-Schublade, wie wenn das Icon auf der Workbench in der Schublade

Prefs angeklickt worden wäre. Dies ist auch dann der Fall, wenn überhaupt kein Argument übergeben wird.

USE/S

Wurde ein Dateiname einer PRINTER-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen aktiviert. Beim Neustart des Rechners gelten jedoch wieder die alten Einstellungen.

SAVE/S

Wurde ein Dateiname einer PRINTER-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen gespeichert und damit bei jedem Neustart des Rechners aktiviert.

PUBSCREEN/K

Wird hier ein Name eines anderen Screens eingegeben, so wird das PRINTER-Fenster auf diesen Screen geöffnet.

UNIT/S

Mit dieser Option kann eine andere Ausgabeschnittstelle ausgewählt werden.

PAG Sandini

PRINTERGFX

Syntax:

2.0: PRINTERGFX [[FROM] <file>] [EDIT] [USE] [SAVE]

2.1/3.0: PRINTERGFX [[FROM] <file>] [EDIT] [USE] [SAVE]
[PUBSCREEN <screen>]

Schablone:

2.0: FROM,EDIT/S,USE/S,SAVE/S

2.1/3.0: FROM,EDIT/S,USE/S,SAVE/S,PUBSCREEN/K

Pfad:

SYS:Prefs

Funktion:

Es kann die Druckeranpassung angezeigt oder verändert werden.

Beschreibung:

Wird PRINTERGFX ohne ein weiteres Argument oder mit dem Schlüsselwort "EDIT" aufgerufen, wird das Programm der Prefs-Schublade gestartet. Wird ein Dateiname angegeben und besitzt diese Datei Informationen zu den Voreinstellungen (sie wurde im PRINTERGFX-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert), können diese schon vorher definierten Einstellungen übernommen werden.

Argumente:

FROM Name einer Datei, die im PRINTERGFX-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert wurde. Was mit diesen bereits definierten Einstellungen passiert, wird mit den anschließenden Schlüsselwörtern festgelegt.

EDIT/S Startet das Programm "PRINTERGFX" der Prefs-Schublade, wie wenn das Icon auf der Workbench in der

Schublade Prefs angeklickt worden wäre. Dies ist auch dann der Fall, wenn überhaupt kein Argument übergeben wird.

USE/S

Wurde ein Dateiname einer PRINTERGFX-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen aktiviert. Beim Neustart des Rechners gelten jedoch wieder die alten Einstellungen.

SAVE/S

Wurde ein Dateiname einer PRINTERGFX-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen gespeichert und damit bei jedem Neustart des Rechners aktiviert.

PUBSCREEN/K

Wird hier ein Name eines anderen Screens eingegeben, so wird das PRINTERGFX-Fenster auf diesen Screen geöffnet.

PAG Sandini

PRINTERPS

Syntax:

2.0: PRINTERPS [[FROM] <file>] [EDIT] [USE] [SAVE]
 2.1/3.0: PRINTERPS [[FROM] <file>] [EDIT] [USE] [SAVE]
 [PUBSCREEN <screen>]

Schablone:

2.0: FROM,EDIT/S,USE/S,SAVE/S
 2.1/3.0: FROM,EDIT/S,USE/S,SAVE/S,PUBSCREEN/K

Pfad:

SYS:Prefs

Funktion:

Es kann die Druckeranpassung angezeigt oder verändert werden.

Beschreibung:

Wird PRINTERPS ohne ein weiteres Argument oder mit dem Schlüsselwort "EDIT" aufgerufen, wird das Programm der Prefs-Schublade gestartet. Wird ein Dateiname angegeben und besitzt diese Datei Informationen zu den Voreinstellungen (sie wurde im PRINTERPS-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert), können diese schon vorher definierten Einstellungen übernommen werden.

Argumente:

FROM	Name einer Datei, die im PRINTERPS-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert wurde. Was mit diesen bereits definierten Einstellungen passiert, wird mit den anschließenden Schlüsselwörtern festgelegt.
EDIT/S	Startet das Programm "PRINTERPS" der Prefs-Schublade, wie wenn das Icon auf der Workbench in der

Schublade Prefs angeklickt worden wäre. Dies ist auch dann der Fall, wenn überhaupt kein Argument übergeben wird.

USE/S

Wurde ein Dateiname einer PRINTERPS-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen aktiviert. Beim Neustart des Rechners gelten jedoch wieder die alten Einstellungen.

SAVE/S

Wurde ein Dateiname einer PRINTERPS-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen gespeichert und damit bei jedem Neustart des Rechners aktiviert.

PUBSCREEN/K

Wird hier ein Name eines anderen Screens eingegeben, so wird das PRINTERPS-Fenster auf diesen Screen geöffnet.

PAG Sandini

PRINTFILES

Syntax:

2.0/2.1/3.0: PRINTFILES [-F] {<file>} [-F|FORMFEED]

Schablone:

2.0/2.1/3.0: -F/S,FILE/A/M,-F=FORMFEED/S

Pfad:

Extras:Tools

Funktion:

Druckt mehrere Textdateien auf dem eingestellten Drucker aus.

Beschreibung:

Mit dem Befehl PRINTFILES können mehrere Textdateien auf dem Drucker ausgegeben werden. Die gleiche Wirkung hat der Befehl "COPY <file> to PRT:". FAS-Sammlung

Argumente:

-F/S Dieser Schalter kann auch am Schluß der Befehlszeile stehen und veranlaßt PRINTFILES, nach jeder ausgedruckten Datei das Papier auf den Beginn der nächsten Seite vorzuschieben. Damit wird beim Ausdrucken mehrerer Dateien verhindert, daß der Beginn einer neuen Datei noch auf der letzten Seite der vorherigen Datei gedruckt wird.

FILE/A/M Ein oder mehrere Namen von Dateien deren Inhalt ausgedruckt werden soll. Die Anzahl der Namen ist lediglich durch die Zeilenlänge von AmigaDOS begrenzt.

Siehe auch:

CMD

PROMPT

Syntax:

2.0/2.1/3.0: PROMT <text>

Schablone:

2.0/2.1/3.0: <text>/A

Pfad:

in die Shell integriert

Funktion:

Bestimmt das Aussehen der Eingabeaufforderung.

Beschreibung:

Der Befehl PROMPT ermöglicht es dem Anwender, die Eingabeaufforderung für den nächsten Befehl nach seinem Geschmack zu gestalten. Dabei können nicht nur Angaben über die Prozessnummer und das aktuelle Verzeichnis gemacht werden, es kann auch ein eigener Text eingebunden oder sogar die Farbe und die Schrift geändert werden.

Es stehen sämtliche Escape-Sequenzen zur Gestaltung zur Verfügung, wenngleich nicht alle sinnvoll sind und einige unzumutbare Resultate liefern. Aber lassen Sie doch einmal Ihrem Entdeckungsdrang freien Lauf, es kann ja nichts passieren.

Der Text der Eingabeaufforderung kann auch drei verschiedene Platzhalter aufnehmen, an deren Stelle bestimmte Informationen eingefügt werden. Diese Platzhalter sind in einzelnen:

%N Fügt an dieser Stelle die Nummer des Prozesses ein.

%S Fügt an dieser Stelle den kompletten Pfad des aktuellen Verzeichnisses ein.

%R An dieser Stelle wird die sogenannte Return-Code (die Fehlernummer)

des zuletzt ausgeführten Befehls angezeigt.

Ganz besondere Möglichkeiten schafft die Verwendung von weiteren Befehlen, wie folgende Beispiele verdeutlichen.

Beispiele:

Probieren Sie doch einmal die Wirkung der unten aufgezeigten Befehle aus:

```
prompt "*E[7mCode: %R*NVerzeichnis: %S*NShellNr.: %N*NBefehl:*E[0m "
```

```
prompt "Verzeichnis: %S Datum: `date` *NBefehl: "
```

```
prompt "*E[33m%S*N%N> *E[0m"
```

Anmerkung:

Alle meine Versuche, Prompt dazu zu bewegen, mehr als nur 80 Zeichen auszugeben, schlugen fehl. Es ist demnach nicht möglich, längere Texte als Eingabeaufforderung zu verwenden. Selbst wenn interaktiv ein Befehl aufgerufen wird, bricht die Eingabeaufforderung nach einer bestimmten Zahl von Zeichen ab. Die Shell ist dann jedoch zur Eingabe weiterer Befehle bereit, so daß mit einem einfachen Befehl

```
PROMPT "%N.%S> "
```

die ursprüngliche Eingabeaufforderung wieder hergestellt werden kann.

Eine Änderung der Eingabeaufforderung wird nur für den Prozess gültig, in dem der Befehl PROMPT eingegeben wurde sowie für alle weiteren davon gestarteten Shells.

Siehe auch:

CD

PROTECT

Syntax:

2.0/2.1/3.0: PROTECT [FILE] <file|muster> [+/- <HSPARWED>]
 [ADD|SUB <HSPARWED>] [ALL] [QUIET]

Schablone:

2.0/2.1/3.0: FILE/A, FLAGS, ADD/S, SUB/S, ALL/S, QUIET/S

Pfad:

C:

Funktion:

PROTECT ändert die Schutzbits von Dateien und Verzeichnissen.

PAG Sandini

Beschreibung:

Zu jeder Datei gehört ein Satz Schutzbits, die gesetzt sind oder nicht. In der Ausgabe des Befehls LIST sind sie ganz deutlich zu sehen. Die Bits und ihre Bedeutung entnehmen Sie bitte folgender Tabelle (die Reihenfolge der Tabelle entspricht der Reihenfolge ihrer Nennung des Befehls LIST):

- A (Archive) Dieses Bit kann dann gesetzt werden, wenn man eine Datei gewissermaßen "versiegeln" will. Diese Bit bleibt solange gesetzt, bis eine Datei verändert wird und bei einer Änderung wird es automatisch gelöscht. An diesem Bit kann man erkennen, ob eine Datei seit der Versiegelung verändert wurde.
- R (Readable) Wenn dieses Bit gesetzt ist, darf die Datei von einem Programm gelesen werden oder der darin enthaltene Text angezeigt werden. Dieses Bit wird oft jedoch mißachtet.
- W (Writeable) Dieses Bit sagt AmigaDOS wenn es gesetzt ist, daß die Datei beschrieben werden darf. Es ist jedoch nicht möglich, die Datei komplett zu löschen, wenn nicht zusätzlich auch das D-Bit gesetzt ist.

- E** (Executable) Es handelt sich um eine Programmdatei, die ausführbar ist.
- D** (Deleteable) Diese Datei darf bei gesetztem Bit gelöscht werden.
- P** (Pure) Dieses Bit ist in Verbindung mit residenten Programmen wichtig. Es sagt AmigaDOS, ob eine Datei resident in den Speicher geladen werden kann (Bit gesetzt) oder nicht (Bit gelöscht). Mehr darüber in der Beschreibung des Befehls RESIDENT.
- S** (Script) Bei dieser Datei handelt es sich um eine Script-Datei. Wird dieses Bit vom Anwender durch den Befehl PROTECT gesetzt, so muß zur Ausführung dieser Befehlsdatei nicht mehr der Befehl EXECUTE vorangestellt werden, es reicht der Name der Datei selbst.
- H** (Hidden) Dieses Bit wird derzeit von AmigaDOS noch nicht unterstützt.

PAG Sandini

Argumente:

- FILE/A** Hier muß der Name der Datei und ein passendes Namensmuster eingegeben werden, deren Schutzbits geändert werden soll. Gehorchen mehr Dateien dem übergebenen Muster, werden bei allen diesen Dateien die Schutzbits verändert. Die Art und Weise, wie sie verändert werden, bestimmen die nächsten Argumente.
- FLAGS** Hier kann der neue Satz Schutzbits eingegeben werden, H, S, P, A, R, W, E, D (Bedeutung siehe oben). Mit den vorangestellten Zeichen + werden die nachfolgenden Bits (auch "flags" genannt) gesetzt, wird ein Minus-Zeichen vorangestellt, so werden die nachfolgend angegebenen Flags gelöscht. Es können mit einem einzigen Befehl gleichzeitig einige Bits gelöscht und dafür andere gesetzt werden.
- ADD/S** Dieser Schalter ist wohl wesentlich verständlicher, als das Zeichen +, hiermit werden die nachfolgend angegebenen Flags gesetzt.

SUB/S Was das Schlüsselwort ADD für das Zeichen “+” ist, stellt SUB für das Minus-Zeichen dar. Die nachfolgenden Bits werden gelöscht.

ALL/S Mit diesem Schalter wird PROTECT dazu überredet, bei allen im übergebenen Verzeichnis vorhandenen Dateien und auch allen in der Verzeichnishierarchie tiefer liegenden Verzeichnissen und Dateien die Bits in der gewünschten Weise zu verändern. Hiermit könnte sogar eine ganze Diskette und alle darauf vorhandenen Verzeichnisse und Dateien angesprochen werden:

1> Protect df0:#? SUB D ALL

QUIET/S Dieser Schalter unterdrückt Meldungen von PROTECT über gesetzte und gelöschte Bits bei den einzelnen Dateien in die Konsole. Eine ähnliche Wirkung hätte die Ausgabeumleitung in das Gerät NIL:, nur werden hier zusätzlich auch alle Fehlermeldungen unterdrückt, die mit QUIET ausgegeben werden würden.

Siehe auch:

LIST

QUIT

Syntax:

2.0/2.1/3.0: QUIT [[RC] <returncode>

Schablone:

2.0/2.1/3.0: RC/N

Pfad:

in die Shell integriert

Funktion:

Bricht die Bearbeitung einer Befehlsdatei mit dem angegebenen Fehlercode ab.

Beschreibung: PAG Sandini

Dieser Befehl kann zwar im Grunde überall eingegeben werden, erfüllt aber nur in einer Befehlsdatei sinnvoll seine Aufgabe. QUIT ermöglicht bei auftreten eines Fehlers einen geordneten Rückzug aus der Schlacht - sprich er bricht die Bearbeitung der Befehlsdatei ab. Zusätzlich kann noch ein sogenannter Returncode übergeben werden, anhand dessen die Schwere des Fehlers (oder der Ort des Auftretens) ermittelt werden kann. Der Befehl QUIT kann in seiner Ausführung nicht unterbrochen werden.

Kommt es in einer Befehlsdatei zur Ausführung des Befehls QUIT (es wird hier angenommen, daß er nur in kritischen Ausnahmefällen ausgeführt werden soll und somit entweder innerhalb eines IF-ELSE-ENDIF- oder eines SKIP-LAB-ENDSKIP-Blocks steht), so werden alle noch offenen IF- oder SKIP-Blöcke ignoriert, und das Script unverzüglich an der Stelle von QUIT verlassen.

Argument:

RC/N Diese Zahl muß nicht unbedingt angegeben werden, sie sollte ausdrücken, wie schwer der aufgetretene Fehler zu bewerten ist. Erlaubt sind eigentlich alle Werte zwischen 0 und

2147483647 (2^31-1), empfohlen werden jedoch (in Analogie zu den Returncodes der normalen AmigaDOS-Befehle) folgende Werte:

- 0 (success) Erfolg
- 5 (warn) Warnung
- 10 (error) Fehler
- 20 (fail) Befehl hat versagt

Bemerkung:

Wird QUIT in einer Befehlsdatei aufgerufen, die selbst von einem anderen Script durch den Befehl EXECUTE gestartet wurde, so wird die Mutter-Datei an der Stelle des Befehls EXECUTE ebenfalls abgebrochen. Wäre die Tochter-Datei jedoch normal beendet worden, so würde die Mutter-Datei nach dem Befehl EXECUTE normal weiterbearbeitet worden. Siehe dazu auch die folgenden Beispiele.

QUIT sollte aber nicht in Befehlsdateien ausgeführt werden, die durch ICONX gestartet werden. Wenn ein solches Script aufgrund eines Fehlers abgebrochen werden soll, so empfiehlt es sich, mit einer SKIP-Anweisung in die allerletzte Zeile des Scripts zu springen, der letzte Befehl dieser Befehlsdatei ist also eine LAB-Anweisung. Stößt nämlich ICONX auf einen QUIT-Befehl, so wird die Bearbeitung wie gewohnt abgebrochen und die Kontrolle des Prozesses wie bei einer Shell an den Anwender gegeben. Es gibt aber vereinzelt Anwender, die sich nur auf der Workbench zuhause fühlen und mit einer Shell nichts anfangen können. Diese Lösung ist für diese Gruppe Amiga-Besitzer nicht optimal, so daß auf jedem Fall versucht werden sollte, eine ICONX-Befehlsdatei normal zu beenden.

Beispiele:

Mit den folgenden Beispielen möchte ich noch einmal die Wirkung von QUIT bei verschachtelten Befehlsdateien zeigen. Dazu wurden zwei unterschiedliche Scripts erstellt, die Mutter-Datei "Script1:"

```
;;Mutter-Datei: ruft die Befehlsdatei Script2 auf
ECHO "Hallo, hier bin ich: Script1"
ECHO "Starte nun Script 2"
EXECUTE Script2
ECHO "Script2 ist nun fertig, hier bin ich wieder, Dein Script1"
```

ruft eine weitere Befehlsdatei "Script2"

;Tochter-Datei "Script2", wird mit QUIT abgebrochen

ECHO "Hallo - hier schreibt Dein Script2 "

ECHO "Hier breche ich ab"

QUIT

ECHO "Abgebrochen, Script2"

auf, in der der Befehl QUIT ausgeführt wird. Beim Aufrufen von Script1 erscheinen folgende Zeilen in der Shell:

1.Ram Disk:> execute script1

Hallo, hier bin ich: Script1

Starte nun Script 2

Hallo - hier schreibt Dein Script2

Hier breche ich ab

Es ist deutlich zu erkennen, das der Befehl QUIT in Script2 auch Script1 abgebrochen hat. Dies läßt sich zwar verhindern, indem Script2 mit den Befehlen RUN EXECUTE als eigener Prozess gestartet wird, doch hat man hier wenig Möglichkeiten um herauszufinden, ob in der zweiten Befehlsdatei ein Abbruch stattfand. Die geänderte Befehlsdatei Script1 sieht wie folgt aus:

;Mutter-Datei: ruft die Befehlsdatei Script2 auf

ECHO "Hallo, hier bin ich: Script1"

ECHO "Starte nun Script 2"

RUN EXECUTE Script2

WAIT 5 SECS ; Warten, ob Script2 abbricht?

ECHO "Script2 ist nun fertig, hier bin ich wieder, Dein Script1"

Wird diese nun gestartet, erhält man folgende Zeilen in der Shell:

1.Ram Disk:> EXECUTE Script1

Hallo, hier bin ich: Script1

Starte nun Script 2

[CLI 5]

Hallo - hier schreibt Dein Script2

Hier breche ich ab

Script2 ist nun fertig, hier bin ich wieder, Dein Script1

Zwar wurde Script2 abgebrochen aber Script1 wurde bis zum Ende ausgeführt. Zusätzlich mußte jedoch in diesem Beispiel durch den Befehl WAIT die Ausführung der Datei Script1 unterbrochen werden, sonst wäre die Ausgabe in der Shell durcheinander geraten (es handelt sich ja um zwei völlig unabhängige Prozesse, die zu unterschiedlichen Zeiten ihre Meldungen ausgeben).

Siehe auch:

FAILAT, IF

PAG Sandini

RELABEL

Syntax:

2.0/2.1/3.0: RELABEL [DRIVE] <gerätname>: [NAME] <neuname>

Schablone:

2.0/2.1/3.0: DRIVE/A, NAME/A

Pfad:

C:

Funktion:

Ändert den Namen eines Datenträgers (Diskette oder Festplattenpartition).

Beschreibung:

Dieser Befehl ändert den Namen einer Diskette oder einer Festplattenpartition. Mehr ließe sich dazu eigentlich nicht sagen, ja wäre da nicht noch eine kleine weitere Feinheit: Der Befehl verlangt vom Anwender, das Gerät anzugeben, in dem die Diskette einliegt, deren Name geändert werden soll. Es gibt aber auch die Möglichkeit, den alten Namen dieser Diskette einzugeben, womit vermieden werden kann, daß irrtümlicherweise eine andere Diskette als die gewünschte umbenannt wird. Bei der Eingabe des Gerätes oder des alten Diskettennamens muß der abschließende Doppelpunkt unbedingt angegeben werden, beim neuen Namen ist er nicht unbedingt erforderlich.

Argumente:

DRIVE/A Dieses Argument muß angegeben werden und sagt dem Befehl, in welchen Gerät eine Diskette umbenannt werden soll oder direkt welche Diskette umbenannt werden soll, falls der noch gültige Diskettenname gefolgt von einem Doppelpunkt eingegeben wird.

NAME/A Auch dieses Argument wird von RELABEL unbedingt gefordert und bezeichnet den neuen Namen der Diskette. Hier ist ein abschließender Doppelpunkt nicht erforderlich. Bei der Wahl des Namens sind die Konventionen von AmigaDOS zu beachten.

Bemerkungen:

Oft wundern sich Anwender, daß sie den Namen geändert haben, die Workbench aber offensichtlich nicht darauf reagiert. Dies liegt daran, daß RELABEL zwar AmigaDOS vom neuen Namen einer eingelegten Diskette informiert, aber nicht die Workbench. Entfernen Sie in diesem Fall die Diskette kurz aus dem Laufwerk und legen Sie sie anschließend wieder ein, dann hat auch die Workbench die Veränderung bemerkt.

Achtung: Bei dem Befehl RELABEL kommt es oft vor, daß der Name der Diskette erst nach einigen Sekunden auf die Diskette geschrieben wird. Warten Sie am besten einige Sekunden, bis das Laufwerk deutlich erkennbar eine Aktion gezeigt hat. Ungeduldige können mit einer Beschädigung der Diskette und einem daraus resultierenden Verlußt von Daten bestraft werden.

Siehe auch:

RENAME, DISKCHANGE, FORMAT

PAG Sandini

REMRAD

Syntax:

2.0/2.1/3.0: REMRAD [[DRIVE] <drive>:] [FORCE]

Schablone:

2.0/2.1/3.0: DRIVE, FORCE/S

Pfad:

C:

Funktion:

Entfernt die resetfeste RAM-Disk und gibt den belegten Speicher wieder frei.

Beschreibung:

Seit der Version 1.3 existiert die resetfeste RAM-Disk "RAD:", die im Gegensatz zu der normalen RAM-DISK "RAM:" auch bei einem Reset des Amigas (beispielsweise über die Tastenkombination Ctrl-Amiga-Amiga) seinen Inhalt nicht verliert, beim Ausschalten des Amigas allerdings schon! Ein weiterer Unterschied besteht darin, daß RAM: nur immer soviel Speicher belegt, wie unbedingt erforderlich ist, RAD: dagegen die Kapazität einer Diskette belegt, auch wenn große Teile des Speichers unbenutzt bleiben. Doch dieses Problem kann kontrolliert werden, da RAD: explizit gestartet werden muß und nicht wie das RAM: automatisch vorhanden ist. Wenn jedoch eine RAD: eingerichtet wird, so ist der reservierte Speicher für das restliche System nicht mehr nutzbar, auch nicht nach einem Reset.

Mit dem Befehl REMRAD kann aber die resetfeste RAM-Disk wieder entfernt und der belegte Speicher freigegeben werden. Ein kleiner Teil jedoch bleibt weiterhin reserviert, der Grund dafür ist in der Arbeitsweise von MO-UNT zu suchen. Nach einem Reset kann aber auch dieser kleine Restteil wieder benutzt werden.

Wird die RAD: jedoch gegenwärtig von einem anderen Prozess genutzt, kann sie nicht mit dem Befehl REMRAD entfernt werden. Dies ist auch dann der Fall, wenn mit ASSIGN ein Verzeichnis innerhalb der RAD: ein logisches

Gerät zugewiesen wurde, ein Verzeichnis der RAD: in einem Suchpfad eines beliebigen Prozesses enthalten oder gar als aktuelles Verzeichnis deklariert ist oder wenn ein Programm von der RAD: aus gestartet wurde, bzw. auf dort gespeicherte Daten zugreift.

Argumente:

DRIVE Hiermit kann angegeben werden, welche RAD: entfernt werden soll. Seit Version 2.0 ist es möglich, mehr als nur eine resetfeste RAM-Disk einzurichten.

FORCE/S Dieses Schlüsselwort zwingt REMRAD dazu, die resetfeste RAM-Disk auch dann zu eliminieren, wenn es eigentlich aus einem der oben genannten Gründen nicht entfernt werden dürfte. Dieses Schlüsselwort ist mit größter Vorsicht anzuwenden. In früheren Versionen (etwa bei 2.0) hatte FORCE die Gestalt eines Schlüsselwortes, dem der Name der zu entfernenden RAD: zugewiesen werden mußte. Die Syntax wäre dann folgende:

1> REMRAD FORCE=RAD:

Siehe auch:

MOUNT, ASSIGN, PATH

RENAME

Syntax:

2.0/2.1/3.0: RENAME [[FROM] {<filelmuster>}] [[TO|AS] <name | verzeichnis>]
[QUIET]

Schablone:

2.0/2.1/3.0: FROM/A/M, TO=AS/A, QUIET/S

Pfad:

C:

Funktion:

Ändert den Namen einer Datei oder eines Verzeichnisses.

Beschreibung:

Wie der Name des Befehls schon erkennen läßt, wird er dazu verwendet, Dateien und Verzeichnisse umzubenennen. Dabei ist zu beachten, daß AmigaDOS keinen Unterschied zwischen Datei- und Verzeichnisnamen macht. Die einfachste Anwendung ist:

1> RENAME Read.Me Lies.Mich ; hier wird eine Datei umbenannt
1> RENAME Bilder Gemälde ; hier wird ein Verzeichnis umbenannt

Aber RENAME hat eine weitere oft vergessene Funktion: Es kann Dateien von einem Verzeichnis in ein anderes verschieben - allerdings nur auf dem gleichen Datenträger. Dies geschieht, indem nicht der Dateiname selbst verändert wird, sondern der Pfadname. Hierbei ändert aber RENAME die Adressen der Datei im sogenannten "file header block", damit die Datei auch wirklich ins neue Verzeichnis kommt. Dies ist auch der Grund, warum eine Datei nicht auf einen anderen Datenträger verschoben werden kann, denn dazu müßten weit größere Eingriffe in die Struktur der Datenträger vorgenommen werden.

Wenn Sie also das Gerät "PC0:" automatisch bei jedem Neustart installieren wollen, brauchen Sie nur folgenden Befehl eingeben:

1> RENAME SYS:Storage/DOSDrivers/PC0#? TO SYS:DEVS/DOSDrivers

Durch die Verwendung der Jokerzeichen wird erreicht, daß gleichzeitig auch das zugehörige Icon mitkopiert wird (sonst wären Inhalte der Schubladen wie sie die Workbench zeigen würde, nicht mehr korrekt). Die Verwendung von Namensmustern wie eben dargestellt kann nur beim Verschieben von Dateien erfolgen. Hierbei muß es sich beim Ziel unbedingt auch um ein richtiges Verzeichnis handeln, das bereits existiert. Denn RENAME kann keine fehlenden Verzeichnisse schaffen. Soll eine richtige Namensänderung vorgenommen werden, sind Jokerzeichen unerwünscht (es könnte ja sonst sein, daß mehrere Dateien den selben Namen bekommen würden - ja und dann....?)

Argumente:

- FROM/A Hier wird der alte Name der Datei oder des Verzeichnisses eingetippt, der geändert werden soll.
- FROM/A/M Seit der Version kann RENAME auch zum Verschieben von Dateien zwischen zwei Verzeichnissen verwendet werden. In diesem Fall können mehrere Dateinamen oder auch Namensmuster als Quelle eingegeben werden. Die angegebenen Dateien oder solche, deren Namen das angegebene Muster enthalten werden dann in das neue Verzeichnis verschoben.
- TO/A, AS/A Beide Schlüsselwörter sind gleichbedeutend. Hinter ihnen wird der neue Name der Datei oder des Verzeichnisses angegeben. Soll RENAME zum Verschieben von Dateien (oder auch Verzeichnissen) eingesetzt werden, so muß hier unbedingt der Name eines schon vorhandenen Verzeichnisses auf dem gleichen Datenträger stehen. RENAME ist weder in der Lage, fehlende Verzeichnisse zu erstellen, noch Dateien zwischen zwei Datenträger zu verschieben.

Anmerkung: Es ist vorteilhaft, wenn in Scriptdateien das Schlüsselwort TO verwendet wird, wenn RENAME Dateien verschieben soll. Wenn aber Namen selbst geändert werden, ist das Schlüsselwort AS eher angebracht. So kann man schon von der Wahl der Schlüsselwörter einen Hinweis auf die Arbeitsweise des Befehls geben.

QUIET/S Ist dieser Schalter gesetzt, so werden alle Meldungen von **RENAME** zum Arbeitsfortgang unterdrückt. Wichtige Fehlermeldungen erscheinen trotzdem.

Bemerkung:

Der kleine aber feine Unterschied zwischen dem Kopieren einer Datei und dem Verschieben besteht darin, daß beim Kopieren die Originaldatei im alten Verzeichnis bestehen bleibt, beim Verschieben aber verschwindet.

Siehe auch:

COPY, DELETE

PAG Sandini

REQUESTCHOISE

Syntax:

2.0/2.1: Befehl nicht implementiert

3.0: REQUESTCHOICE [TITLE] <titel> [BODY] <meldungstext>
<schaltertext> [PUBSCREEN <screen>]

Schablone:

2.0/2.1: Befehl nicht implementiert

3.0: TITLE/A, BODY/A, GADGETS/A/M, PUBSCREEN/K

Pfad:

C:

Funktion: PAG Sandini

Erstellt einen Requester nach den Vorgaben des Anwenders und teilt die Reaktion mit.

Beschreibung:

Dieser Befehl ist für Programmierer eine unsagbare Hilfe, stellt er doch dem armen Kerl auf überraschend unbürokratische Weise einen Requester zur Verfügung, für den in früheren Betriebssystemversionen viele Nerven und viel Zeit verloren gingen. In Befehlsdateien gab es überhaupt keine Möglichkeit, den Anwender über Requester zwischen verschiedenen Aktionen entscheiden zu lassen. Hier mußten komplizierte Konstruktionen mit den Befehlen ECHO, ASK, IF (bei Fortgeschrittenen evtl. auch TYPE, SKIP und LAB) auch auf die merkwürdigsten Antworten des verwirrten Anwenders vorbereitet sein.

Die Zeiten haben sich geändert, dem Luxus der neuen Workbench 3.0 muß man Dank und Anerkennung schenken. Das Erstellen eines Requesters ist denkbar einfach: Sie geben an, welcher Text im Fenster erscheinen soll, welche Gadgets zur Verfügung gestellt werden sollen und der Befehl REQUESTCHOICE gibt die Nummer des ausgewählten Gadgets zurück.

Da der Befehl die Nummer des Ausgewählten Gadgets normalerweise direkt ausgibt, muß die Antwort über eine Ausgabeumleitung in eine Env:-Variable umgeleitet werden, um eine Auswertung zu ermöglichen. Dabei ist zu beachten, daß das äußerst rechte Gadget immer die Nummer 0 hat (es ist für eine Antwort wie "CANCEL" oder frei übersetzt etwa "vergib es" gedacht, bei der keine Reaktion erfolgen soll). Alle anderen Gadgets werden links mit 1 beginnend in Einserschritten durchnummeriert. Die maximale Anzahl der Gadgets ist durch die Breite des Bildschirms bzw. der Textlänge der einzelnen Gadgets begrenzt. Es kann nur eine Reihe Gadgets zur Verfügung gestellt werden.

Argumente:

- | | |
|-------------|---|
| TITLE/A | Dieser Text muß angegeben werden, er erscheint im Titel des Requester-Fensters. |
| BODY/A | Dieser Text stellt die Frage dar, die so formuliert werden sollte, daß sie klar durch eines der angebotenen Gadgets beantwortet werden kann. Es gibt die Möglichkeit, bei längeren Texten durch die Zeichen "*n" einen Zeilenumbruch vorzugeben. REQUESTCHOICE verlangt eine Frage, wird sie vergessen, so interpretiert der Befehl den Titel des ersten Gadgets als Frage. |
| GADGETS/M | Hier werden nun die Titel der einzelnen Gadgets eingegeben. Die Namen sollten so gewählt werden, daß die ausgelöste Reaktion klar ersichtlich ist und die Frage eindeutig beantwortet werden kann. Die Anzahl der Gadgets ist lediglich durch die Breite des Bildschirms begrenzt (und auch durch die Länge der AmigaDOS-Befehlszeile). Wird ein Gadget gewählt, so gibt REQUESTCHOICE die Nummer des entsprechenden Gadgets aus. Die Gadgets werden von links nach rechts mit 1 beginnend fortlaufend durchnummeriert mit einer Ausnahme: das Gadget ganz rechts hat immer die Nummer 0. |
| PUBSCREEN/K | Hiermit kann REQUESTCHOICE veranlaßt werden, den Requester auf einem anderen Screen als der Workbench |

zu öffnen. Dieser Screen wird automatisch in den Vordergrund geholt, sobald der Requester erscheint, nach seiner Beantwortung wird der ursprünglich im Vordergrund dargestellte Screen wieder nach vorne geholt.

Beispiel:

Dieses Beispiel erfüllt nur einen Zweck, es soll die Arbeitsweise und Auswertung des Befehls REQUESTCHOICE verdeutlichen. Dazu habe ich folgende Befehlsdatei namens "ReqChoice.Demo" erstellt:

```
REQUESTCHOICE >ENV:Antwort "Diese ist nur eine Demonstration!"  
"Welches Scheinder'l hätten's den gern?" rot gelb grün blau  
IF $Antwort EQ 0  
ECHO "Das blaue bitte!"  
ENDIF  
IF $Antwort EQ 1  
ECHO "Das rote bitte!"  
ENDIF  
IF $Antwort EQ 2  
ECHO "Das gelbe bitte!"  
ENDIF  
IF $Antwort EQ 3  
ECHO "Das grüne bitte!"  
ENDIF
```

Wird bei der Ausführung das zweite Gadget von links ausgewählt, so erscheint die Antwort:

```
1.Ram Disk:> execute reqchoice.demo  
Das gelbe bitte!
```

Einfacher geht's wohl nicht mehr!

Siehe auch:

REQUESTFILE

REQUESTFILE

Syntax:

2.0/2.1: Befehl nicht implementiert

3.0: REQUESTFILE [[DRAWER} <verzeichnis> [FILE <dateiname>]
[PATTERN <muster>] [TITLE <titel>] [POSITIVE <pos.text>]
[NEGATIV <neg.text>] [ACCEPTPATTERN <muster>]
[REJECTPATTERN <muster>] [SAVEMODE] [MULTISELECT]
[DRAWERONLY] [NOICONS] [PUBSCREEN <screen>]

Schablone:

2.0/2.1: Befehl nicht implementiert

3.0: DRAWER, FILE/K, PATTERN/K, TITLE/K, POSITIVE/K,
NEGATIVE/K, ACCEPTPATTERN/K, REJECTPATTERN/K,
SAVEMODE/S, MULTISELECT/S, DRAWERONLY/S,
NOICONS/S, PUBSCREEN/K

Pfad:

C:

Funktion:

Erstellt einen Dateiauswahlrequester und gibt die ausgewählte Datei als Antwort zurück.

Beschreibung:

Auch wenn ich oft behaupte, daß das Arbeiten mit der Shell und der Tastatur wesentlich mehr Freiheiten bietet und ein schnelleres Arbeiten ermöglicht, und ich daher die Maus fast nicht benutze, muß ich dennoch eines eingestehen: Einer der wesentlichen Gründe, warum ich nicht den MEMACS zum erstellen des Manuskripts dieses Buches verwende, sondern einen nahen Verwandten dieses Editors, ist der daß MEMACS keine Dateiauswahl-Requester unterstützt. Diese geniale Erfindung erleichtert die Arbeit mit dem Computer in unschätzbare Weise. Um so erfreulicher ist es, daß AmigaDOS durch den Befehl REQUESTFILE einen sehr einfach einzusetzenden Filerequester zur Verfügung stellt.

Nach soviel Lob auf den Filerequester jetzt aber zu den eigentlichen Errungenschaften: Mit dem Befehl REQUESTFILE kann ein sehr komfortabler Dateiauswahl-Requester erstellt und eingesetzt werden. Dabei kann REQUESTFILE noch eine Reihe verschiedener Vorgaben berücksichtigen, so daß der Requester universal einsetzbar wird aber auch an ganz spezifische Probleme angepaßt werden kann. Die Anpassungen finden über die Argumente statt, die im folgenden beschrieben werden.

Argumente:

- | | |
|------------|--|
| DRAWER | Wird REQUESTFILE kein Argument übergeben, so liest der Dateiauswahl-Requester automatisch das aktuelle Verzeichnis. Mit dieser Option kann aber auch jedes beliebige andere Verzeichnis angegeben werden, dessen Inhalt von Beginn an im Requester dargestellt wird. |
| FILE/K | Soll schon eine bestimmte Datei voreingestellt werden, so muß hier das Schlüsselwort FILE gefolgt vom Namen der Datei angegeben werden. Diese Datei gilt dann bereits als ausgesucht, die Auswahl kann direkt durch Drücken auf das OK-Gadget erfolgen. |
| PATTERN/K | Wird hier ein Namensmuster angegeben, so erscheint im File-Requester ein Gadget "Pattern", in dem dieses Muster wieder zu finden ist. Es werden in diesem Fall nur die Namen der Dateien gezeigt, die zu dem Muster passen. |
| TITLE/K | Ab und an ist es sehr nützlich, dem Fenster eigenen Namen zu geben, um zu verdeutlichen, was mit der ausgewählten Datei wirklich geschieht. Um den Fensternamen zu ändern, muß der neue an das Schlüsselwort TITLE anschließen. |
| POSITIVE/K | Hiermit kann dem linken Gadget ein anderer Name als das voreingestellte "OK" gegeben werden. |
| NEGATIVE/K | Hiermit legen Sie den Namen des rechten Gadgets fest, das normalerweise "Cancel" heißt. |

ACCEPTPATTERN/K Auch hier kann ein Muster für die Dateien angegeben werden, deren Namen im Requester dargestellt werden, wenn sie zu diesem Muster passen. Anders als bei **PATTERN** erscheint hierbei jedoch kein "Pattern"-Gadget.

REJECTPATTERN/K Mit dem hier angegebenen Namensmuster verhält es sich genau umgekehrt wie bei **ACCEPTPATTERN**. Alle Dateien, deren Namen dem angegebenen Muster gehorchen, werden nicht dargestellt. Auch hier erscheint kein "Pattern"-Gadget im Requester.

SAVEMODE/S Wird dieser Schalter angegeben, so erscheinen sämtliche Namen im Dateiauswahlrequester invertiert. Dies fällt als erstes ins Auge. Es gibt aber noch eine weitere Besonderheit:

Es ist in diesem Modus nicht möglich, eine Datei über einen Doppelklick auf den Namen auszuwählen, sondern hierzu muß die Auswahl explizit mit dem Gadget **OK** bestätigt werden. Dieses Argument eignet sich hervorragend, wenn mit den ausgewählten Dateien gravierende Änderungen vorgesehen sind, oder diese gelöscht werden.

MULTISELECT/S Durch diesen Schalter wird **REQUESTFILE** in einen Modus gesetzt, in dem mehr als nur eine Datei ausgewählt werden kann. Die Auswahl weiterer Dateien erfolgt wie in der Workbench durch gedrückt halten einer Shift-Taste und gleichzeitigem Anklicken der Dateinamen. Es können jedoch immer nur ein Verzeichnis Dateien ausgewählt werden.

Wird das Verzeichnis gewechselt, so werden die schon ausgewählten Dateien deselektiert (zu deutsch: vergessen). Die Namen der ausgewählten Dateien erscheinen in der Ausgabe jeweils in Anführungszeichen eingeschlossen und durch Leerzeichen voneinander getrennt.

- DRAWERONLY/S** Mit diesem Schalter kann REQUESTFILE dazu veranlaßt werden, nur Verzeichnisse zu zeigen und alle darin enthaltenen Dateien wegzulassen.
- NOICONS/S** Mit dieser Option werden alle .info-Dateien, die für die Icons auf der Workbench verantwortlich sind, von REQUESTFILE nicht angezeigt.
- PUBSCREEN/K** Wie schon so oft, kann auch REQUESTFILE dazu gebracht werden, das Fenster des Dateiauswahlrequesters in einem anderen Screen zu öffnen. Diese Möglichkeit könnte unter ARexx sehr sinnvoll eingesetzt werden.

Siehe auch:
REQUESTCHOICE

PAG Sandini

RESIDENT

Syntax:

2.0/2.1/3.0: **RESIDENT** [[NAME] <res.name>] [[FILE] <programmname>]
 [REMOVE] [ADD] [REPLACE] [PURE|FORCE] [SYSTEM]

Schablone:

2.0/2.1/3.0:
 NAME, FILE, REMOVE/S, ADD/S, REPLACE/S, PURE=FORCE/S, SYSTEM/S

Pfad:

in die Shell integriert

Funktion:

Die Liste der resistenten Programme wird angezeigt oder verändert.

Beschreibung: PAG Sandini

Der Befehl **RESIDENT** wurde zum ersten Mal auf der Workbench 1.3 eingesetzt. So unscheinbar dieser Befehl auch aussieht, er hat das AmigaDOS seither wesentlich beeinflusst. Im Grunde ermöglicht dieser Befehl nur, AmigaDOS-Befehle und andere Programme, die bestimmte Anforderungen erfüllen, ständig lauffähig im RAM zu halten. Damit entfallen bei jedem Aufruf des residenten Befehles die sonst anfallenden Ladezeiten.

Wer jetzt argumentiert, daß es doch schon bei der ersten Workbench möglich war, Befehle in die RAM-Disk zu kopieren und von dort aus zu starten, der weiß offensichtlich noch nicht, daß diese Befehle trotzdem bei jedem Aufruf eine (wenn auch deutlich geringere) Ladezeit benötigen. Die Programme in der RAM-Disk sind in der vorliegenden Form noch nicht ausführbar. Hinzu kommt, daß das Programm jedesmal neu in den Speicher geladen werden muß, wenn mehrere Prozesse diesen einen Befehl gleichzeitig benutzen. Werden die Programme jedoch mit **RESIDENT** im RAM abgelegt, so stehen diese sofort ausführbereit zur Verfügung. Dies bedeutet, daß ein residentes Programm nur ein einziges Mal im Speicher steht, selbst wenn es zur gleichen Zeit von mehreren Prozessen ausgeführt. Hieraus ergibt sich auch gleich die notwendige Voraussetzung, die ein Programm erfüllen muß, um resident

geladen werden zu können: Es muß so programmiert sein, daß ein einziger Programm-Code von mehreren Prozessen gleichzeitig benutzt werden kann, ohne daß sich Variablen oder Adressen, die in den einzelnen Prozessen natürlich unterschiedlich sind, gegenseitig beeinflussen. Programme, die diese Voraussetzung erfüllen werden im Fachjargon "pure" genannt. Durch das gleichnamige Schutzbit "PURE" (siehe auch PROTECT) wird dem System angezeigt, daß das Programm die gestellten Anforderungen erfüllt und somit resident in den Speicher geladen werden kann. Man sollte sich davor hüten, ein fehlendes PURE-Bit zu setzen, wenn das Programm resident geladen werden soll, die Voraussetzungen aber nicht erfüllen kann - Guru läßt grüßen.

Wird der Befehl RESIDENT ohne weiteren Argumente aufgerufen, so wird nur die Liste der derzeit residenten Befehle gezeigt. Hinter den einzelnen Namen steht eine Spalte namens "Use Count", in der die Anzahl gezeigt wird, welcher Befehl derzeit von wievielen Prozessen bearbeitet wird. Zumindest sollte es so sein, aber wie folgendes Beispiel zeigt, ist auch bei der Zeile Resident keine "1", sondern auch hier steht das Wort "INTERNAL".

1. Work:DOS3.0> resident

NAME	USE COUNT
------	-----------

Alias	INTERNAL
Ask	INTERNAL
CD	INTERNAL
Echo	INTERNAL
Else	INTERNAL
EndCLI	INTERNAL
EndIf	INTERNAL
EndShell	INTERNAL
EndSkip	INTERNAL
Failat	INTERNAL
Fault	INTERNAL
Get	INTERNAL
Getenv	INTERNAL
If	INTERNAL
Lab	INTERNAL
NewCLI	INTERNAL
NewShell	INTERNAL

Path	INTERNAL
Prompt	INTERNAL
Quit	INTERNAL
Resident	INTERNAL
Run	INTERNAL
Set	INTERNAL
Setenv	INTERNAL
Skip	INTERNAL
Stack	INTERNAL
Unalias	INTERNAL
Unset	INTERNAL
Unsetenv	INTERNAL
Why	INTERNAL
.ket	INTERNAL
.bra	INTERNAL
.key	INTERNAL

Argumente: PAG Sandini

- NAME** Normalerweise wird der residente Befehl unter dem gleichen Namen eingetragen, wie auch auf dem Massenspeicher, von wo er resident geladen wird. Hiermit kann aber der Name des Befehls in der Resident-Liste geändert werden. Wird eine Änderung des Namens gewünscht, so muß der neue Name unbedingt vor den Namen des Befehls stehen, der resident gemacht werden soll.
- FILE** Hier wird der Name des Befehls angegeben, der resident gemacht werden soll. Wird vorher kein anderer Name übergeben (siehe Argument NAME), so wird der Befehl unter dem gleichen Namen resident gehalten.
- REMOVE/S** Mit diesem Schalter wird der angegebene bereits residente Befehl wieder aus der Liste entfernt. Hierbei ist der Name einzugeben, der in der Liste erscheint, sollte er sich vom eigentlichen Namen des Befehls unterscheiden. Anmerkung: Es ist nicht möglich, voreingestellte interne Befehle zu entfernen, es würde ohnehin kein zusätzlicher

Speicher dadurch verfügbar werden. Um einen internen Befehl zu entfernen, muß entweder ADD oder REPLACE benutzt werden. Ein damit entfernter Befehl wird in der Liste mit "DISABLED" gekennzeichnet.

ADD/S

Mit diesem Schlüsselwort wird RESIDENT gesagt, daß der neue Befehl in die gültige Liste hinzugefügt wird. Dies wird vorallem verwendet, um beispielsweise in Befehlsdateien vorübergehend Befehle resident zu machen und hinterher wieder zu entfernen, ohne daß andere vom Benutzer resident geladenen Befehle beim Entfernen beeinträchtigt werden.

REPLACE/S

Der mit diesem Schalter gewählte Modus ist eigentlich schon voreingestellt, so daß dieses Schlüsselwort nicht notwendig ist. RESIDENT versucht immer einen bereits residenten Befehl durch den angegebenen gleichen Namens zu ersetzen. Ist kein solcher Eintrag vorhanden, wird der Befehl hinzugefügt.

PURE=FORCE/S

Dieser Schalter muß mit größter Vorsicht eingesetzt werden. PURE oder FORCE zwingt den Befehl RESIDENT dazu, Programme resident zu laden, auch wenn das Pure-Bit (siehe Befehl PROTECT) nicht gesetzt ist - womit das Programm sehr wahrscheinlich nicht die notwendigen Voraussetzungen erfüllt. Der Befehl RESIDENT gibt in diesem Fall zwar eine Warnung aus, führt aber seine Arbeit fort. Es kann zum Absturz des Rechners führen, wenn ein nicht geeignetes Programm resident gemacht wird und von mehreren Prozessen gleichzeitig aufgerufen wird.

SYSTEM/S

Dieses Schlüsselwort kann zwei verschiedene Auswirkungen haben, je nachdem welche anderen Umstände zutreffen: 1. Wird RESIDENT nur mit dem Schlüsselwort SYSTEM als einziges Argument aufgerufen, so zeigt der Befehl die Liste aller internen Befehle und anschließend die Liste der Einträge seitens des Betriebssystems.

2. Wird dieses Schlüsselwort in Verbindung mit den Namen eines resident zu ladenden Befehles verwendet, so wird dieser in die Liste seitens des Betriebssystems aufgenommen. Diese Verwendung sollte Programmentwicklern vorbehalten bleiben, normale Amiga-Programme (dazu gehören auch alle AmigaDOS-Befehle) sollten auf keinen Fall auf diese Weise resident gemacht werden.

Bemerkungen:

Die Programmierung von Befehlen, die resident geladen werden können, stellt die Software-Entwickler immer wieder vor große Probleme. Denn wenn ein Programm "pure" sein soll, muß es "reentrant" oder "wiedereintrittsfähig" und auch wiederholt ausführbar "reexecutable" sein. Dies erfordert eine sehr exakte Programmierung nach bestimmten komplizierten Spielregeln.

Für den Anwender ist eigentlich nur wichtig zu wissen, ob ein Programm diese Anforderungen erfüllt. Normalerweise sollte dann das Schutzbit "PURE" gesetzt sein, aber beim Kopieren kann ab und zu schon einmal ein Schutzbit verloren gehen. Doch gibt es auch eine Möglichkeit, durch Versuche herauszufinden, ob ein bestimmtes Programm resident geladen werden kann.

a) Prüfen, ob ein Programm wieder ausführbar ist: Dieser Test ist der wichtigste und sollte zuerst durchgeführt werden. Zuerst muß ermittelt werden, ob ein Programm mehr als nur einmal mit dem gleichen Programm-Code ausgeführt werden kann, ohne daß sich die einzelnen Prozesse gegenseitig beeinflussen. Dazu sind folgende Schritte notwendig:

1. Zuerst muß der Befehl resident gemacht werden unter Verwendung des Schlüsselwortes PURE.
2. Starten Sie das Programm mit allen möglichen Argumenten. Anschließend verlassen Sie das Programm wieder.
3. Starten Sie das Programm erneut. Sollte jetzt der Amiga abstürzen oder das Programm andere Krankheiten zeigen, kann es mit Sicherheit nicht resident geladen werden. Scheint jedoch alles in Ordnung zu sein, kann es möglicherweise endgültig in die Liste der residenten Befehle aufgenommen werden. Programme, die vom CLI aus gestartet werden können,

sollten zuerst einmal nur mit dem unbedingt notwendigen Minimum an Argumenten gestartet werden. Es sollten dann alle möglichen Kombinationen der Argumente (v.a. der Schlüsselwörter) wie folgt ab Schritt 2 noch einmal geprüft werden. Sollte dabei auch nur ein einziges Mal ein merkwürdiges Verhalten auffallen, so ist der Befehl wohl ungeeignet, also nicht "pure".

b) Probe, ob der Befehl reentrant ist: Auch dieser Test ist wichtig, da hiermit ermittelt wird, ob ein Programm mit nur einem Code gleichzeitig von mehreren Prozessen abgearbeitet werden kann. Hat ein Programm zwar den ersten Test (a) erfüllt, aber zeigt hier Schwächen, kann es dennoch resident geladen werden, jedoch unter der Einschränkung, daß es niemals von zwei verschiedenen Prozessen gleichzeitig aufgerufen wird. Die Vorgehensweise im einzelnen:

1. Wie bei Test a) muß das Programm mit dem Schlüsselwort **FORCE** resident geladen werden.
2. Nun muß es mit dem Befehl **RUN** mehrmals gleichzeitig gestartet werden. Ist der Befehl ein reiner **CLI**-Befehl, so ist diese Anforderung nicht ganz leicht zu erfüllen. Es gibt zwei ähnliche Möglichkeiten: Unter Verwendung des Befehls **RUN** unter Anfügen des "+"-Zeichens an den Befehlsnamen:

```
RUN test+  
test+  
test
```

In diesem Fall wartet **RUN**, bis einmal kein "+"-Zeichen mehr auftritt, bevor alle Befehle (dann gleichzeitig) gestartet werden. Die zweite Möglichkeit ist das Erstellen einer Befehlsdatei, in dem das Programm mehrmals hintereinander mit **RUN** gestartet wird. Allerdings sollte hierbei darauf geachtet werden, daß keine **SKIP-LAB-ENDSKIP**-Blöcke vorhanden sind.

3. Prüfen Sie, ob sich die einzelnen Prozesse gegenseitig beeinflussen. Manchmal ist dies durch Auswahl eines Menüpunktes erkennbar, wenn der zweite Prozess auf die Auswahl des ersten reagiert. Allerdings sind große Programme mit vielen Menüs, Gadgets, Fenster usw. ohnehin

schlecht geeignet, resident geladen zu werden. Wenn keine gegenseitige Behinderung erkennbar ist, so ist das Programm "pure". (Ausnahmen kann es aber immer geben - lassen Sie sich überraschen!)

Befehle, die das Betriebssystem ändern (wie etwa SETPATCH), sollten nicht resident geladen werden, da sie so programmiert sind, daß Sie nur einmal beim Systemstart aufgerufen werden.

Alle Befehle in diesem Buch, bei denen zum Pfad "in die Shell integriert" angegeben wurde, stehen in der Liste der residenten Programme.

Siehe auch:

PROTECT, WHICH

PAG Sandini

REXXMAST

Syntax:

2.0/2.1/3.0: REXXMAST

Schablone:

2.0/2.1/3.0: -

Pfad:

Sys:System

Funktion:

Startet den ARexx-Interpreter.

Beschreibung:

Der Befehl REXXMAST startet den ARexx-Interpreter als Hintergrundprozess. Dieses Programm muß unbedingt im Speicher sein, damit ARexx-Programme ausgeführt werden können. Wird der Befehl REXXMAST eingegeben, so installiert er automatisch den Interpreter und übergibt anschließend die Kontrolle wieder an AmigaDOS:

```
1.Work:DOS3.0> rexxmast  
ARexx Version 1.15  
Copyright © 1987 by William S. Hawes  
All Rights Reserved
```

Wird der Befehl ein weiteres Mal eingegeben, so erscheint folgende Meldung:

```
1.Work:DOS3.0> rexxmast  
ARexx Version 1.15  
Copyright © 1987 by William S. Hawes  
All Rights Reserved  
REXX server already active
```

Eine Überprüfung schadet nicht, aber ARexx-Programme arbeiten ohne den Interpreter nicht. Das Programm benötigt nicht sehr viel Speicher, so daß es bei intensiver Arbeit mit ARexx möglicherweise von Vorteil ist, wenn das Programm bereits in der Befehlsdatei "S:User-Startup" gestartet wird. Es gibt auch Versionen, in denen dies bereits in der "S:Startup-Sequence" geschieht. Wenn der Interpreter entfernt werden soll, muß RXC (steht für "RexX Close") eingegeben werden.

RUN

Syntax:

2.0/2.1/3.0: RUN <command> [+ {<command>}]

Schablone:

2.0/2.1/3.0: COMMAND/F

Pfad:

in die Shell integriert

Funktion:

Startet einen oder mehrere Befehle als selbständigen Prozess.

Beschreibung:

In vielen Dokumentationen wurde geschrieben, daß RUN andere Befehle als Hintergrundprozess startet. Wenn ich jedoch mit dem Multitasking arbeite und beispielsweise mehrere Kopiervorgänge gleichzeitig mit RUN startet, so habe ich oft nicht den Eindruck, daß diese Programme im Hintergrund laufen, sondern sich eher in den Vordergrund drängen, vor allem wenn sie Bildschirmausgaben vornehmen. Eine treffendere Beschreibung ist wohl, daß ein Programm als selbstständiger Prozess gestartet wird und gleichberechtigt nebenher läuft.

Im Prinzip öffnet RUN wie auch NEWCLI eine eigene Shell jedoch ohne Fenster. Alle Ein- und vor allem Ausgaben erfolgen über das Shell-Fenster in dem der RUN-Befehl eingetippt wurde, es sei denn, die Ein- oder Ausgaben wurden umgeleitet (mit ">" oder "<"). Wie bei dem Befehl NEWCLI übernimmt auch RUN das aktuelle Verzeichnis, die Priorität und die Größe des Stapelspeichers von der startenden Shell.

Der Befehl, der als eigener Prozess laufen soll, wird hinter dem Befehl RUN genauso eingegeben, wie wenn er direkt in der Shell laufen würde. Alle notwendigen Argumente werden genauso übergeben. Beispielsweise ermöglicht folgende Zeile, ein ganzes Verzeichnis zu kopieren, während in der Shell weitergearbeitet werden kann:


```
1> RUN COPY df0:C/#? TO RAM:C ALL QUIET [CLI 4]
```

In eckigen Klammer wird die Nummer des neuen Prozesses angezeigt, über die beispielsweise der Befehl **BREAK** eingreifen kann. Auch für andere Befehle wie **CHANGETASKPRI** oder **STATUS** ist diese Nummer wichtig. Welche Nummer dieser neue Prozess zugeteilt bekommt, hängt davon ab, wieviel andere Prozesse bereits gestartet wurden und noch aktiv sind. Mit dem Befehl **STATUS** kann auch ständig überwacht werden, ob ein Prozess noch aktiv ist oder nicht.

Da viele Befehle die **QUIET**-Option nicht unterstützen, aber eine Ausgabe in das Shell-Fenster bei Verwendung von **RUN** unerwünscht ist, kann die Ausgabe mit ">NIL:" unterbunden werden. Es besteht hier jedoch ein wichtiger Unterschied:

Bei Angabe von **QUIET** erscheinen etwaige Fehlermeldungen oder Warnungen trotzdem auf dem Bildschirm, bei einer Ausgabeumleitung ins Nichts, werden jedoch auch diese unterschlagen, womit man keine entgeltliche Kontrolle mehr hat. Hier könnte eine Ausgabe in eine Datei im **RAM:** abhelfen, da hier nachträglich Meldungen gelesen werden können, die in der Shell stören würden.

Mit **RUN** können auch Befehlsdateien über den Befehl **EXECUTE** oder bei bereits gesetztem "Script"-Bit direkt über den Dateinamen als selbständige Prozesse gestartet werden.

Sollen mit einem **RUN**-Befehl mehrere Programme gestartet werden, die nacheinander auszuführen sind, so können diese durch ein Pluszeichen "+" voneinander getrennt in der entsprechenden Reihenfolge hinter **RUN** angegeben werden.

Da oft die Argumente der einzelnen Befehle so lang sind, wartet **RUN** auch nach Drücken der **RETURN**-Taste, wenn das letzte Zeichen der Zeile ein Plus "+" war. In diesem Fall wird die nächste und alle weiteren Zeilen als weitere Befehle akzeptiert, die in der angegebenen Reihenfolge als eigenständiger Prozess abgearbeitet werden.

Beispiel:

Das folgende Beispiel zeigt zum einen, daß **RUN** wartet, bis wirklich eine Zeile ohne einem "+"-Zeichen endet, bevor ein neuer Prozess gestartet wird:

```
1.Work:DOS3.0> RUN COPY dos? to df0: quiet +  
dir df0: +  
echo "Fertig"  
[CLI 5]  
1.Work:DOS3.0> status  
Process 1: Loaded as command: status  
Process 2: Loaded as command: C:ConClip  
Process 3: Loaded as command: hd0:m2emacs  
Process 4: Loaded as command: sys:tools/commodities/blanker  
Process 5: Loaded as command: COPY  
1.Work:DOS3.0> dos1 dos2  
dos3 dos4  
dos5 workbench  
Fertig  
1.Work:DOS3.0>
```

Unter dem Prozess Nummer 5 werden dann die eingegebenen Befehle ausgeführt, was auch der Befehl **STATUS** bestätigt. Wie lästig Bildschirmausgaben durch einen mit **RUN** gestarteten Befehl sein können, wird hierbei obendrein auch noch deutlich.

Siehe auch:

STATUS, CHANGETASKPRI, NEWCLI

SAY

Syntax:

2.0/2.1: SAY [<text>] [-M|-F|-R|-N] [-S=<n>] [-P=<n>] [-X=<datei>]

3.0: Befehl nicht implementiert

Schablone:

2.0/2.1: TEXT, -M/S, -F/S, -R/S, -N/S, -S/K/N, -P/K/N, -X/K

3.0: Befehl nicht implementiert

Pfad:

Sys:Utilities

Funktion:

Läßt den Amiga einen Text sprechen.

Beschreibung:

Mit diesem Befehl kann ein Text über den Sprachsynthesizer des Amigas ausgegeben werden. Ohne irgendwelche Argumente wirkt dieser Befehl, wie wenn das gleichnamige Icon in der Utilities-Schublade der Workbench aufgerufen worden wäre. Damit läßt sich dann sehr schön mit verschiedenen Werten herumexperimentieren. In der Shell kann die Anwendung von SAY oft sehr unterhaltsam sein - etwa wenn der Amiga Sie lautstark willkommen heißt.

Argumente:

TEXT Der Text, der gesprochen werden soll (da der Sprachsynthesizer vor allem an die englische Aussprache angelehnt ist, werden deutsche Wörter oft stark verfälscht wiedergegeben).

-M/S Dieser Schalter ist voreingestellt und sagt dem Amiga, er soll eine männliche Stimme haben.

-F/S Mit diesem Argument ertönt eine weibliche Stimme.

- R/S Hier spricht ein Roboter (monoton).
- N/S Natürliche Betonung (voreingestellt).
- S/K/N Hier kann die Sprechgeschwindigkeit in Wörter je Minute eingestellt werden. Die gültigen Werte reichen von 40 bis 400.
- P/K/N Dieser Wert gibt die Tonhöhe des Grundtones an. Die gültigen Werte reichen von 65 bis 320.
- X/K Hiermit kann der Namen einer Datei angegeben werden, die der Befehl dann vorliest.

Bemerkung:

Ich gebe hier bewußt keine Beispiele an, da dieser Befehl zum Experimentieren sehr geeignet ist.

PAG Sandini

SCREENMODE

Syntax:

2.0/2.1/3.0: SCREENMODE [[FROM] <file>] [EDIT] [USE] [SAVE]

Schablone:

2.0/2.1/3.0: FROM,EDIT/S,USE/S,SAVE/S

Pfad:

SYS:Prefs

Funktion:

Es kann die Bildschirmdarstellung angezeigt und verändert werden.

Beschreibung:

Wird SCREENMODE ohne ein weiteres Argument oder mit dem Schlüsselwort "EDIT" aufgerufen, wird das Programm der Prefs-Schublade gestartet. Wird ein Dateiname angegeben und besitzt diese Datei Informationen zu den Voreinstellungen (sie wurde im SCREENMODE-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert), können diese schon vorher definierten Einstellungen übernommen werden.

Argumente:

FROM Name einer Datei, die im SCREENMODE-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert wurde. Was mit diesen bereits definierten Einstellungen passiert, wird mit den anschließenden Schlüsselwörtern festgelegt.

EDIT/S Startet das Programm "SCREENMODE" der Prefs-Schublade, wie wenn das Icon auf der Workbench in der Schublade Prefs angeklickt worden wäre. Dies ist auch dann der Fall, wenn überhaupt kein Argument übergeben wird.

- USE/S** Wurde ein Dateiname einer SCREENMODE-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen aktiviert. Beim Neustart des Rechners gelten jedoch wieder die alten Einstellungen.
- SAVE/S** Wurde ein Dateiname einer SCREENMODE-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen gespeichert und damit bei jedem Neustart des Rechners aktiviert.

PAG Sandini

SEARCH

Syntax:

2.0/2.1/3.0: SEARCH [[FROM] {<filelmuster>}] [[SEARCH] <stringlmuster>]
 [ALL] [NONUM] [QUIET] [QUICK] [FILE] [PATTERN]

Schablone:

2.0/2.1/3.0: FROM/M, SEARCH/A, ALL/S, NONUM/S, QUIET/S, QUICK/S,
 FILE/S, PATTERN/S

Pfad:

C:

Funktion:

Durchsucht Dateien nach Zeichenketten oder Verzeichnisse nach Dateien.

Beschreibung:

Mit dem Befehl SEARCH können Dateien nach Zeichenketten durchsucht werden. Es ist auch möglich, ganze Datenträgern nach Dateien zu durchforsten. Dabei müssen die Schlüsselwörter FROM und SEARCH nicht notwendigerweise angegeben werden, verbessern aber vor allem in Befehlsdateien die Übersichtlichkeit. Es können beliebige Dateien, Namen oder Namensmuster bei FROM angegeben werden nach dem Auftreten einer Zeichenkette (diese oder ein Muster davon wird bei SEARCH eingetippt) durchsucht. Jedesmal wenn Search auf eine derartige Zeichenkette stößt, wird die Zeilennummer gefolgt von der Zeile in das Fenster ausgegeben. Im folgenden Beispiel wird in der Startup-Sequence nach dem Auftreten der beiden Zeichen "IF" gesucht. Der Befehl SEARCH findet dazu:

```
1.Work:DOS3.0> SEARCH FROM S:Startup#? SEARCH IF
Startup-Sequence..
24 IF NOT EXISTS SYS:Fonts
26 EndIF
31 IF EXISTS DEVS:Monitors
32 IF EXISTS DEVS:Monitors/VGAOnly
34 EndIF
```

```

39 EndIF
52 IF EXISTS S:User-Startup
54 EndIF

```

Darüber hinaus gibt es auch die Möglichkeit, daß die Zeichenkette, nach der gesucht wird, als Name oder entsprechendes Namensmuster einer Datei interpretiert wird, indem der Schalter **FILE** gesetzt wird. Wenn Sie also auf der Workbench alle Dateien suchen, die mit dem Wort "Amiga" beginnen, geben sie folgendes ein:

```

DOS3.0> SEARCH FROM dh0: SEARCH Amiga#? FILE ALL
Work:DH0/sonstiges/basic/AmigaBASIC
Work:DH0/sonstiges/basic/AmigaBASIC.info
Work:DH0/Amiga-Seminar/Amiga-Seminar/Amiga-Einsteiger1
Work:DH0/Amiga-Seminar/Amiga-Seminar/Amiga-Einsteiger2
Work:DH0/TeX/inputs/amigauml.sty
Work:DH0/TeX/inputs/amigauml.tex
Work:DH0/TeX/TeX/mfinputs/ini/amiga.mf
Work:DH0/TeX/TeX/src/mf/MFwindow/amiga.c
Work:DH0/Modula-2/m2-Arbeitsspeicher/funktion/AmigaFunktion.def
Work:DH0/Modula-2/m2-Arbeitsspeicher/funktion/AmigaFunktion.mod

```

Dieser Befehl ist gerade für Besitzer von Festplatten eine große Hilfe, da oft eine längst verschollen geglaubte Datei dadurch wieder aufgefunden werden kann.

Argumente:

FROM Hier kann die Quelle angegeben werden, die nach der zu suchenden Zeichenkette durchkämmt wird. Es sind auch Namensmuster oder auch Namen von Verzeichnissen erlaubt. Letzere kommen eher dann in Frage, wenn nach Dateien selbst gesucht wird. Werden keine Verzeichnisnamen übergeben, so wird im aktuellen Verzeichnis gesucht.

SEARCH/A Diese Zeichenkette, nach der gesucht werden soll, muß natürlich angegeben werden, sonst kann **SEARCH** nichts anfangen. Das Schlüsselwort selbst muß nicht angegeben werden, verbessert aber vor allem in Script-Dateien die

Lesbarkeit. Es ist auch hier möglich, Muster für die zu suchende Zeichenkette einzugeben, allerdings nur dann, wenn nach einer Datei gesucht wird und nicht eine Zeichenkette innerhalb von Dateien. Mit Hilfe des Schlüsselwortes PAT-TERN ist es auch möglich, eine Zeichenkette innerhalb einer Datei durch Jokerzeichen zu beschreiben.

ALL/S Wird dieses Schlüsselwort benutzt, so sucht Search nicht nur im angegebenen Verzeichnis, sondern auch in allen weiteren darin enthaltenen Unterverzeichnissen. Normalerweise kann mit Namensmustern nur eine Ebene abgehandelt werden.

NONUM/S Normalerweise stellt SEARCH die Zeilennummer der Zeile voran, in der die zu suchende Zeichenkette aufgetreten ist. Mit diesem Schlüsselwort kann jedoch die Ausgabe der Zeilennummer unterdrückt werden.

QUIET/S Dieser Schalter ist nur bei Einsatz des Befehls SEARCH in Befehlsdateien sinnvoll, da es jegliche Ausgabe des Befehls in das Fenster unterdrückt. Wird das zu suchende Objekt nicht gefunden, wird eine Warnung (Fehlercode 5) erzeugt, war die Suche erfolgreich, so ist der Fehlercode 0. Damit ist es möglich, das Resultat mit IF-ELSE-ENDIF-Blöcken auszuwerten.

QUICK/S Mit dieser Option wird die Ausgabe von SEARCH ein wenig abgeändert, so daß dadurch etwas Zeit gespart werden kann. Normalerweise zeigt SEARCH seinen Arbeitsfortgang an, indem alle Dateien ausgegeben werden, die gerade durchsucht werden. Ist jedoch dieser Schalter gesetzt, so wird nicht mehr für jede durchsuchte Datei eine neue Zeile begonnen, sondern nur dann, wenn in der letzten Datei die gesuchte Zeichenkette aufgetreten ist. Siehe dazu auch nachfolgende Beispiele.

FILE/S Mit diesem Schlüsselwort wird dem Befehl SEARCH mitgeteilt, daß er nicht Dateien nach der angegebenen Zeichenkette durchsuchen soll, sondern Verzeichnisse nach einer Datei,

deren Namen mit der Zeichenkette identisch ist, oder dem angegebenen Namensmuster gehorcht. Normalerweise wird dieser Schalter in Verbindung mit ALL benutzt.

PATTERN/S Mit diesem Schlüsselwort wird SEARCH darauf aufmerksam gemacht, daß die in der Zeichenkette auftretende Jokerzeichen auch als solche zu bewerten sind und nicht direkt als Zeichen Bestandteil der zu findenden Zeichenkette in Dateien sein sollen. Damit können Jokerzeichen auch für Zeichenketten in Dateien selbst eingesetzt werden.

Beispiele:

Die folgenden beiden SEARCH-Befehle wurden mit den gleichen Argumenten aufgerufen und durchsuchten das exakt gleiche Verzeichnis, der einzige Unterschied bestand darin, daß einmal das Schlüsselwort QUICK zusätzlich angegeben wurde:

1. Work:DOS3.0> SEARCH FROM s:#? SEARCH #? ALL

DPat..

Ed-startup..

PCD..

SPat..

Startup-Sequence..

29 C:Mount>NIL: DEVS:DOSDrivers/~(#?.info)

36 C:List>NIL: DEVS:Monitors/~(#?.info|VGAOnly) TO T:M LFOR-
MAT "DEVS:Monitors/%s"

Shell-Startup..

BRUtab..

HDBackup.config..

User-Startup..

User-Startup1..

1. Work:DOS3.0> SEARCH FROM s:#? SEARCH #? ALL QUICK
Workbench:S/Startup-Sequence

29 C:Mount>NIL: DEVS:DOSDrivers/~(#?.info)

36 C:List>NIL: DEVS:Monitors/~(#?.info|VGAOnly) TO T:M LFOR-
MAT "DEVS:Monitors/%s"

Durch die Angabe des Schlüsselwortes PATTERN werden Jokerzeichen auch in Zeichenketten akzeptiert und nicht nur in Namensmuster für Dateien:

```
1.Work:DOS3.0> SEARCH FROM S: SEARCH !?b ALL PATTERN QUICK
Workbench:S/DPat
27 Lab doit
Workbench:S/Startup-Sequence
21 Assign >NIL: LIBS: SYS:Classes ADD
```

Wird das Schlüsselwort PATTERN nicht angegeben, so entsteht folgende Ausgabe:

```
1.Work:DOS3.0> SEARCH FROM S: SEARCH !?b ALL QUICK
1.Work:DOS3.0>
```

SEARCH findet also keine Zeichenkette "!?b".

Anmerkung:

Wie aus den obigen Beispielen schon ersichtlich ist, unterscheidet SEARCH keine Groß- und Kleinbuchstaben.

Siehe auch:

WHICH

SERIAL

Syntax:

2.0: SERIAL [[FROM] <file>] [EDIT] [USE] [SAVE]

2.1/3.0: SERIAL [[FROM] <file>] [EDIT] [USE] [SAVE]
[PUBSCREEN <screen>] [UNIT]

Schablone:

2.0: FROM,EDIT/S,USE/S,SAVE/S

2.1/3.0: FROM,EDIT/S,USE/S,SAVE/S,PUBSCREEN/K,UNIT/S

Pfad:

SYS:Prefs

Funktion:

Es können die Einstellungen für die Serielle Schnittstelle angezeigt oder verändert werden.

Beschreibung:

Wird SERIAL ohne ein weiteres Argument oder mit dem Schlüsselwort "EDIT" aufgerufen, wird das Programm der Prefs-Schublade gestartet. Wird ein Dateiname angegeben und besitzt diese Datei Informationen zu den Voreinstellungen (sie wurde im SERIAL-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert), können diese schon vorher definierten Einstellungen übernommen werden.

Argumente:

FROM Name einer Datei, die im SERIAL-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert wurde. Was mit diesen bereits definierten Einstellungen passiert, wird mit den anschließenden Schlüsselwörtern festgelegt.

EDIT/S	Startet das Programm "SERIAL" der Prefs-Schublade, wie wenn das Icon auf der Workbench in der Schublade Prefs angeklickt worden wäre. Dies ist auch dann der Fall, wenn überhaupt kein Argument übergeben wird.
USE/S	Wurde ein Dateiname einer SERIAL-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen aktiviert. Beim Neustart des Rechners gelten jedoch wieder die alten Einstellungen.
SAVE/S	Wurde ein Dateiname einer SERIAL-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen gespeichert und damit bei jedem Neustart des Rechners aktiviert.
PUBSCREEN/K	Wird hier ein Name eines anderen Screens eingegeben, so wird das SERIAL-Fenster auf diesen Screen geöffnet.
UNIT/S	Mit dieser Option kann eine andere Ausgabeschnittstelle ausgewählt werden.

SET

Syntax:

2.0/2.1/3.0: SET [[NAME] <variable>] [String]

Schablone:

2.0/2.1/3.0: NAME,STRING/F

Pfad:

in die Shell integriert

Funktion:

Weist einer lokalen Variable einen Wert zu.

Beschreibung:

Der Befehl SET dient zur Definition einer lokalen (Umgebungs-)Variable. Es gibt wesentliche Unterschiede zwischen globalen und lokalen Umgebungsvariablen, es muß jedoch darauf hingewiesen werden, daß eine lokale Variable einer gleichnamigen globalen stets vorrangig ist. Lokale Variablen sind fest an den Prozess gebunden, in dem sie definiert wurden, wird dieser Prozess beendet, geht ihr Inhalt verloren.

Wird jedoch von einer Shell aus ein neuer Prozess gestartet, so übernimmt dieser alle Definitionen von lokalen Variablen auch in den neuen Prozess. Diese sind von da an jedoch getrennt, so daß der Inhalt einer Variabel des ersten Prozesses geändert werden kann, ohne daß sich der Inhalt der gleichnamigen Variable des zweiten Prozesses ebenfalls ändert und umgekehrt. Sollen jedoch Variablen auch von anderen Prozessen aus erreichbar sein, so müssen sie global definiert werden (durch Verwendung des Befehls SET ENV).

Der Wert einer lokalen Variable kann mit dem Befehl GET ausgelesen werden, gelöscht wird sie mit dem Befehl UNSET. In Script-Dateien werden lokale Variablen genauso angesprochen wie globale, durch voranstellen des Dollarzeichens an den eigentlichen Variablennamen.

Wird der Befehl SET ohne weitere Argumente eingegeben, so zeigt er alle definierten lokale Variablen an:

```
1.Work:DOS3.0> set
```

```
process      1
```

```
RC           0
```

```
Result2      0
```

Hier sind drei besondere lokale Variablen aufgelistet, da diese vom Betriebssystem automatisch definiert werden und beispielsweise auch nicht mit dem Befehl UNSET entfernt werden können. Die Variable "process" enthält immer die Nummer des Shell-Prozesses, in RC steht der Fehlercode (auch ReturnCode genannt) des letzten ausgeführten Befehls, also 0, 5, 10 oder 20. Dieser Wert könnte auch in der Eingabeaufforderung erscheinen, indem in der Zeichenkette des Befehls PROMPT der Platzhalter %R gesetzt wird. Die dritte Variable namens "Result2" enthält unter Umständen weitere Details zum aufgetretenen Fehler (meist, wenn RC einen Wert über 10 besitzt).

PAG Sandini

Argumente:

NAME Der Name der Variablen, der der Wert zugewiesen werden soll.

STRING/F Eine Zeichenkette, die der Variablen als Inhalt zugewiesen wird. Da diese Zeichenkette sowieso die Eingabe des Befehls SET abschließt, sind in diesem Fall keinen Anführungszeichen notwendig, auch wenn die Zeichenkette Leerzeichen enthalten würde.

Siehe auch:

GET, SETENV, UNSET, UNSETENV

SETCLOCK

Syntax:

2.0/2.1/3.0: SETCLOCK <LOAD|SAVE|RESET>

Schablone:

2.0/2.1/3.0: LOAD/S, SAVE/S, RESET/S

Pfad:

C:

Funktion:

Stellt die akkugepufferte Echtzeituhr.

Beschreibung:

Die meisten Amigas - mit Ausnahme des Amiga 1200 - verfügen über akkugepuffert Echtzeituhren, die die Uhr und das Datum auch dann beibehalten, wenn der Amiga abgeschaltet ist. Der Akku wird immer automatisch geladen, wenn der Amiga eingeschaltet ist. Der Befehl SETCLOCK schreibt die aktuelle Zeit und das eingestellte Datum des Amigas in diese Echtzeituhr. Vorher muß also mit dem Befehl DATE die Zeit eingestellt - oder zumindest überprüft- werden.

SETCLOCK unterstützt folgende Schlüsselwörter, die sich jedoch gegenseitig ausschließen:

Argumente:

LOAD/S Hier soll sich das Betriebssystem die aktuelle Uhrzeit von der akkugepufferten Echtzeituhr holen und fortan benutzen. Normalerweise wird dieser Befehl schon in der Startup-Sequence ausgeführt.

SAVE/S Hier wird die aktuelle Systemzeit in die akkugepufferte Uhr übernommen. Dies kann auch auf einem zweiten Weg erreicht werden, indem im Preferences-Programm "Time" das Save-Gadget benutzt wird.

RESET/S

Dieses Schlüsselwort wird benötigt, wenn irgend ein anderes Programm (oder etwa ein Virus) die akkugepufferte Uhr mit Unsinn beschrieben oder angehalten hat. Unter normalen Bedingungen wird dieses Schlüsselwort nicht benötigt.

Anmerkung:

Die Uhr des Betriebssystems und die akkugepufferte Uhr sind zwei völlig voneinander unabhängige Uhren. Wird mit dem Befehl DATE die interne Uhr geändert, so ändert sich die Zeit der akkugepufferten Echtzeituhr nicht automatisch mit. Erst durch Anwendung des Befehls SETCLOCK wird die neue Zeit in die Akku-Uhr übernommen.

Siehe auch:

DATE, SETDATE

PAG Sandini

SETDATE

Syntax:

2.0: SETDATE [FILE] <filelmuster> [[DATE] <datum>]
[[TIME] <zeit>]

2.1/3.0: SETDATE [FILE] <filelmuster> [[WEEKDAY] <wochentag>]
[[DATE] <datum>] [[TIME] <zeit>] [ALL]

Schablone:

2.0: FILE/A, DATE, TIME

2.1/3.0: FILE/A, WEEKDAY, DATE, TIME, ALL/S

Pfad:

C:

Funktion:

PAG Sandini

Ändert das Datum und die Uhrzeit einer Datei.

Beschreibung:

Mit diesem Befehl kann der "Datumsstempel" einer Datei, der beispielsweise bei dem Befehl LIST benutzt wird, geändert werden. Wie der Befehl DATE kann auch SETDATE Namen von Wochentagen verarbeiten, je nach der eingestellten Sprache etwa "Gestern" oder "Samstag" (oder in Englisch "Yesterday" oder "Saturday"). Auch das Format für die Uhrzeit und das Datum entspricht dem des Befehl DATE, die Uhrzeit wird in der Form HH:MM:SS oder HH:MM (ohne Sekundenangabe) eingegeben, das Datum als TT-MMM-JJ oder TT-MM-JJ.

Bei den Zahlen müssen immer zwei Ziffern eingegeben werden, sollte eine Zahl nur einziffrig sein, muß eine Null vorangestellt werden, andernfalls kann SETDATE nichts damit anfangen. Beim Datum ist noch zu bemerken, daß der Monatsname mit den ersten drei Buchstaben anzugeben ist oder alternativ die zweistellige Nummer des Monats. Wird SETDATE keine Zeit übergeben, so wird die akute Zeit der entsprechenden Datei zugewiesen.

SETDATE beeinflusst die akkugepufferte Echtzeituhr und die interne Uhr nicht.

Argumente:

FILE/A

Der Befehl SETDATE benötigt unbedingt den Namen oder Namensmuster der Dateien, dessen Zeit geändert werden soll. Befinden sich die betreffenden Dateien nicht im aktuellen Verzeichnis, so muß der komplette Pfad übergeben werden. Hinweis: SETDATE bildet eine Ausnahme, da die Angabe eines logischen Geräte (etwa "S:") nicht automatisch das Muster #? setzt, es muß explizit angegeben werden. Nebenbei bemerkt, es kann nie schaden, wenn man zusätzlich zu den logischen Gerät immer die Zeichen #? angibt, wenn alle Dateien in diesem Verzeichnis behandelt werden sollen. Siehe dazu auch das untenstehende Beispiel.

DATE

Hier kann das Datum angegeben werden, das bei den entsprechenden Dateien gesetzt wird. Bei der Workbench 1.2 mußte dieses Datum explizit angegeben werden. In allen späteren Versionen ist es erlaubt, das Datum auch wegzulassen, in diesem Fall wird das aktuelle Datum gesetzt. Wird keine Uhrzeit angegeben, so wird die des Amigas für die Datei gestellt. Es ist nicht möglich, nur das Datum neu zu setzen, ohne daß die Uhr verändert wird.

TIME

Hiermit wird die zu setzende Zeit festgelegt. Wird nur eine Uhrzeit angegeben und das Datum weggelassen, so ändert SETDATE nicht nur die Uhr, sondern das Datum wird auf das des Amigas gestellt. Die Zeit alleine kann also nicht verstellt werden.

ALL/S

Mit diesem Schlüsselwort können alle Dateien in einem Verzeichnis und den weiteren Unterverzeichnissen mit einem einzigen Befehl behandelt werden.

WEEKDAY

Hier kann der Name des Wochentages wie etwa "Sonntag" oder "Mittwoch") aber auch relative Tagangaben wie beispielsweise "gestern" oder "morgen" ange-

geben werden. Welche Namen akzeptiert werden, hängt von der eingestellten Sprache ab, die mit dem Preferences-Programm "Locale" gesetzt wurde.

Beispiel:

Das folgende Beispiel zeigt, daß SETDATE bei einem Gerät unbedingt die Jokerzeichen "#?" benötigt, sonst können nicht alle darin enthaltenen Dateien nicht bearbeitet werden:

1.Work:DOS3.0> list ram:

Directory "ram:" on Sunday 21-May-78

User-Startup1	1058	—rwed Yesterday 23:26:56
User-Startup1O	1032	—rwed Yesterday 23:26:56
User-Startup	201	—rwed Yesterday 23:26:56
User-StartupO	94	—rwed Yesterday 23:26:56
Shell-Startup	118	-s—rw-d Yesterday 23:26:56
Startup-Sequence	1213	-s—rw-d Yesterday 23:26:56
Ed-startup	615	—rw-d Yesterday 23:26:56
ENV	Dir	—rwed Yesterday 23:26:56
Clipboards	Dir	—rwed Yesterday 23:26:56
T	Dir	—rwed Yesterday 23:26:56

7 files - 3 directories - 23 blocks used

1.Work:DOS3.0> setdate ram: today

SetDate failed: object is not of required type

1.Work:DOS3.0> setdate ram:#? today

1.Work:DOS3.0> list ram:

Directory "ram:" on Sunday 21-May-78

User-Startup1	1058	—rwed Today 23:27:17
User-Startup1O	1032	—rwed Today 23:27:17
User-Startup	201	—rwed Today 23:27:17
User-StartupO	94	—rwed Today 23:27:17
Shell-Startup	118	-s—rw-d Today 23:27:17
Startup-Sequence	1213	-s—rw-d Today 23:27:17
Ed-startup	615	—rw-d Today 23:27:17
ENV	Dir	—rwed Today 23:27:17
Clipboards	Dir	—rwed Today 23:27:17
T	Dir	—rwed Today 23:27:17

7 files - 3 directories - 23 blocks used

Wird beispielsweise nur

1> Setdate ram:

eingetippt, so kann der Befehl nichts damit anfangen, es erscheint die Meldung "object is not of required type".

Siehe auch:

DATE, SETCLOCK

PAG Sandini

Beispiele:

Durch den folgenden Befehl werden die Anführungszeichen in den Inhalt übernommen:

```
1.Work:DOS3.0> setenv ausgabe echo "Dies ist nur ein kleiner Test"
```

```
1.Work:DOS3.0> run $ausgabe
```

```
Dies ist nur ein kleiner Test
```

```
[CLI 5]
```

```
1.Work:DOS3.0>
```

Das Beispiel zeigt, daß auch Befehle als Variable gespeichert und mit dem Befehl RUN ausgeführt werden können.

Siehe auch:

GET, GETENV, SET, UNSET, UNSETENV

PAG Sandini

SETFONT

Syntax:

2.0/2.1/3.0: SETFONT [NAME] <name> <size> [SCALE] [PROP] [ITALIC]
[BOLD] [UNDERLINE]

Schablone:

2.0/2.1/3.0: NAME/A, SIZE/A/N, SCALE/S, PROP/S, ITALIC/S, BOLD/S,
UNDERLINE/S

Pfad:

C:

Funktion:

SETFONT ändert den Zeichensatz der Shell.

PAG Sandini

Beschreibung:

Seit der Workbench 2.0 existiert die Möglichkeit, mit Hilfe des Befehls SETFONT die Schriftart der Shell zu ändern. Es wird aber nur die Schrift des aktuellen Fensters geändert. Mit dem Programm "Font" aus der Preferences-Schublade können nur die Schriftarten der Workbench festgelegt werden, die Shell benutzt normalerweise weiterhin den altbewährten "Topaz 8"-Zeichensatz.

Der Befehl SETFONT benötigt mindestens zwei weitere Angaben, den Namen des Zeichensatzes und die Schriftgröße in Bildpunkten. Dabei erwartet SETFONT, daß der genannte Zeichensatz im Verzeichnis des logischen Gerätes FONTS: vorhanden ist, womit eine Angabe des kompletten Pfades normalerweise unnötig wird. Befindet sich der Zeichensatz jedoch in einem anderen Verzeichnis, kann entweder durch eine zeitweilige Änderung der FONTS:-Zuweisung oder durch eine vollständige Pfadangabe auch ein anderer Zeichensatz verwendet werden.

Wird der Befehl SETFONT benutzt, so wird der Inhalt des aktuellen Fensters gelöscht und die Eingabeaufforderung in der neuen Schriftart dargestellt.

Argumente:

NAME/A	Dieses Argument muß angegeben werden und bezeichnet den Namen des gewünschten Zeichensatzes. Befindet sich der entsprechende Zeichensatz nicht im Verzeichnis FONTS:, so muß entweder diese Zuweisung geändert, oder ein kompletter Pfadname eingegeben werden.
SIZE/A/N	Auch dieses Argument muß angegeben werden, es gibt die Größe der gewünschten Schrift in Bildpunkten an. SETFONT ist jedoch nicht in der Lage, fehlende Schriftgrößen zu erstellen, es können nur bereits vorhandene Größen benutzt werden. Sie sollten daher darauf achten, daß der Zeichensatz die angegebene Größe besitzt. Ist die gewünschte Größe nicht vorhanden, so wird eine benachbarte Größe benutzt.
SCALE/S	Mit diesem Schalter kann SETFONT auch Schriftarten, die nicht in der gewünschten Größe vorliegen aber skalierbar sind, selbst skalieren. Es können Größen zwischen 1 und 125 Punkte angegeben werden, ein Punkt entspricht einer Höhe von 1/72 Zoll.
PROP/S	Dieser Schalter muß gesetzt werden, wenn sogenannte proportionale Schriften verwendet werden sollen. Dabei handelt es sich um Zeichensätze, bei denen die Buchstaben keine einheitliche Breite besitzen. Von den bei der Workbench 3.0 mitgelieferten Zeichensätzen sind alle bis auf "Topaz" und "Courier" proportionale Zeichensätze. Lesen dazu auch die unten genannten Bemerkungen.
ITALIC/S	Mit diesem Schalter werden die Buchstaben kursiv dargestellt. Für eine Ausgabe in ein Fenster mag es ja ganz schön aussehen, zum Eingeben ist es jedoch denkbar ungeeignet.
BOLD/S	Diese Option läßt die Schrift in Fettdruck erscheinen.

UNDERLINE/S

Und hiermit kann die Schrift unterstrichen werden. Der optische Eindruck ist jedoch nicht allzu berauschend.

Bemerkungen:

Prinzipiell können mit dem Befehl SETFONT sämtliche mitgelieferten Schriftarten in der Shell benutzt werden, in der Praxis zeigt sich jedoch, daß lediglich die beiden nicht proportionalen Zeichensätze vernünftig verarbeitet werden. Bei den proportionalen Schriftarten treten Probleme auf, sobald eine Eingabezeile geändert werden soll, da die Position des optischen Cursors nicht mehr mit der Position im Speicher übereinstimmt. Der Cursor wird nicht an die unterschiedlichen Breiten der einzelnen Buchstaben angepaßt. So kann es schon einmal vorkommen, daß Sie das fünf Zeichen von rechts löschen wollen und stattdessen 7 gelöscht haben. Gerade bei brenzlichen Befehlen wie etwa DELETE oder FORMAT kann dies gravierende Konsequenzen haben.

Da sich jedoch generell über den Geschmack streiten läßt, und das Experimentieren doch sehr unterhaltsam ist, sollten Sie die einzelnen Schriftarten ruhig einmal austesten. Hüten Sie sich jedoch davor, während des Tests kritische Befehle mit möglicherweise verheerenden Folgen zu verwenden.

Siehe auch:

ASSIGN, FIXFONTS

SETKEYBOARD

Syntax:

2.0: Befehl nicht implementiert

2.1/3.0: SETKEYBOARD [KEYMAP] <tastaturbelegung>

Schablone:

2.0: Befehl nicht implementiert

2.1/3.0: KEYMAP/A

Pfad:

C:

Funktion:

Ändert die Tastaturbelegung.

Beschreibung:

Dieser Befehl macht eigentlich genau das gleiche wie sein Vorgänger SETMAP, er ändert die aktuelle Tastaturbelegung. Es sind auf den Systemdisketten folgende Tastaturen verfügbar:

cdn	Kanada (französisch)
ch1	Schweiz (französisch)
ch2	Schweiz (deutsch)
d	Deutsch
dk	Dänisch
e	Spanisch
f	Französisch
gb	Großbritannien (Englisch)
i	Italienisch
n	Norwegisch
po	Portugiesisch
s	Schwedisch
usa0	Amerikanisch (für uralte Programme)
usa	Amerikanisch (Englisch)
usa2	Dvorak

Mit dem Voreinstellungsprogramm "IPREFS" kann die Tastatur ebenfalls bestimmt werden, der Befehl SETKEYBOARD dient eher der kurzfristigen Änderung.

Argumente:

KEYMAP/A Hier muß eines der oben angegebenen Kennzeichen angegeben werden. Befindet sich die Tastaturbelegung nicht im Verzeichnis KEYMAPS:, so muß ein vollständiger Pfad angegeben werden.

Siehe auch:

SETMAP

PAG Sandini

SETMAP

Syntax:

2.0: SETMAP [KEYMAP] <tastaturbelegung>

2.1/3.0: Befehl nicht implementiert

Schablone:

2.0: KEYMAP/A

2.1/3.0: Befehl nicht implementiert

Pfad:

C:

Funktion:

Ändert die Tastaturbelegung.

RAG Sandini

Beschreibung:

Da der Amiga in verschiedene Länder ausgeliefert wird, die verschiedene Standard-Belegungen der Tastatur besitzen, ist es sehr Vorteilhaft, wenn jeder Benutzer die Standard-Belegung benutzen kann, die er von seiner Schreibmaschine gewohnt ist. Es werden sich bestimmt schon viele Anwender gefragt haben (oder sogar darüber geärgert haben), warum denn manchmal die Z-Taste ein Y produziert und umgekehrt. Der Doppelpunkt ist nicht mehr da, wo er sein sollte usw.

Die Begründung dafür ist eigentlich einsichtig. Der Amiga benutzt nach jedem Neustart die amerikanische Tastaturbelegung "usa", die sich eben in diesen auffallenden Punkten von der deutschen DIN-Tastatur unterscheidet. Durch den Befehl SETMAP D kann jedoch die deutsche Tastaturbelegung hergestellt werden, bei der dann alle Zeichen dort erscheinen, wo sie auf den Tasten stehen.

Normalerweise wird diese Tastaturanpassung in der Startup-Sequence vorgenommen, sodaß der Benutzer diese Änderung gar nicht wahrnimmt. Die dafür notwendigen Informationen befinden sich in den Dateien, die bei der

Beschreibung zum Befehl SETKEYMAP aufgezählt sind und normalerweise im Verzeichnis DEVS:Keymaps zu finden sind. Da aber auf den Systemdisketten oft zuwenig Speicherplatz vorhanden ist, werden die Dateien auf die Storage-Diskette oder Extras-Diskette ausgelagert und müssen daher von dort erst einmal auf die Boot-Diskette kopiert werden. Es reicht, eine einzige Tastaturbelegung zu kopieren, jede weitere würde unnötig Platz verschwenden.

Normalerweise sucht SETMAP im Verzeichnis DEVS:Keymaps nach der angegebenen Tastatur, es ist jedoch auch möglich, einen Pfad in ein anderes Verzeichnis einzugeben, wenn sich dort die gewünschte Tastatur befindet.

Argument:

KEYMAP/A Hier muß eines der oben angegebenen Kennzeichen angegeben werden. Befindet sich die Tastaturbelegung nicht im Verzeichnis KEYMAPS:, so muß ein vollständiger Pfad angegeben werden.

Siehe auch:

SETKEYBOARD

PAG Sandini

SETPATCH

Syntax:

2.0: SETPATCH [QUIET] [NOCACHE]

2.1/3.0: SETPATCH [QUIET] [NOCACHE] [REVERSE]

Schablone:

2.0: QUIET/S, NOCACHE/S

2.1/3.0: QUIET/S, NOCACHE/S, REVERSE/S

Pfad:

C:

Funktion:

Korrigiert mögliche Fehler des Betriebssystems.

Beschreibung:

So sorgfältig die Entwickler des Amigas auch arbeiten, es ist ein Ding der Unmöglichkeit zu verlangen, daß das Betriebssystem (auch der Teil im ROM) frei von jeglichen Fehlern sein soll. Mittlerweile ist der Teil im ROM immerhin auf satte 512 KByte angewachsen, Fehler können nicht ausgeschlossen werden. Hierzu wurde der Befehl SETPATCH geschaffen, der Adressen im ROM ändert, damit die Betriebssystemroutinen korrekt arbeiten.

Sobald ein weiterer Fehler entdeckt wird, wird der Befehl SETPATCH erweitert, damit auch dieser Fehler beseitigt werden kann. Dabei sollte der Befehl SETPATCH nicht als Mangel angesehen werden, er ist vielmehr ein großer Vorteil für den Anwender, da damit auch nachträglich gefundene Fehler nachträglich korrigiert werden können. Es gibt viele andere Computerhersteller, die in diesem Fall entweder ein neues (aber vermutlich auch nicht hundertprozentig fehlerfreies) ROM herausgeben oder aber sagen: "Sie müssen eben versuchen, mit dem Fehler klar zu kommen!" Commodore dagegen versucht, nach jeder Entdeckung eines neuen Fehlers, diesen sofort über eine Software zu beheben, bevor es einen neuen ROM-Chip entwickelt.

Existiert ein Befehl SETPATCH, so sollte dieser bei jedem Neustart ausgeführt werden, diese Arbeit nimmt die originale Startup-Sequence dem Anwender bereits ab. Wird dann ein weiteres Mal versucht, diesen Befehl auszuführen, so wird mit einer Meldung und dem Fehlercode 5 (WARN) darauf hingewiesen, daß er bereits schon einmal aufgerufen wurde und ein weiterer Aufruf unnötig war.

Argumente:

- QUIET/S** Hiermit wird SETPATCH zum Schweigen gezwungen, sämtliche Ausgaben unterbleiben.
- NOCACHE/S** Durch dieses Schlüsselwort wird der Befehlpuffer für die Prozessoren 68030 und 68040 ausgeschaltet.
- REVERSE/S** Dieses Schlüsselwort wird in erster Linie von Entwicklern für CDTV-Programme benutzt und reserviert Speicher aus dem oberen Teil des RAM.

Siehe auch:

CPU

PAG Sandini

SHOWCONFIG

Syntax:

- 2.0: Befehl nicht implementiert
- 2.1/3.0: SHOWCONFIG [DEBUG]

Schablone:

- 2.0: Befehl nicht implementiert
- 2.1/3.0: DEBUG/K

Pfad:

SYS:Utilities

Funktion:

Zeigt Informationen über die Konfiguration (Zusammensetzung) des Amigas an.

Beschreibung:

Der Befehl wird normalerweise ohne Argumente verwendet und liefert bei einem Amiga 1200 folgende Ausgabe:

```
1.Work:DOS3.0> showconfig
PROCESSOR: CPU 68020
CUSTOM CHIPS: AA PAL Alice (id=$0023), AA Lisa (id=$00F8)
VERS: Kickstart version 39.106, Exec version 39.47, Disk version 39.29
RAM: Node type $A, Attributes $703 (CHIP), at $400-$1FFFFFF (~2.0 meg)
BOARDS:
None
```

Argumente:

- DEBUG/K Wird dieser Schalter gesetzt, so werden automatisch eingebundene Erweiterungskarten nicht mit gesonderten DEBUG-Informationen bedacht. Die Ausgabe bleibt jedoch gleich:

AMIGA DOS 3.0

1.Work:DOS3.0> showconfig debug

PROCESSOR: CPU 68020

CUSTOM CHIPS: AA PAL Alice (id=\$0023), AA Lisa (id=\$00F8)

VERS: Kickstart version 39.106, Exec version 39.47, Disk version 39.29

RAM: Node type \$A, Attributes \$703 (CHIP), at \$400-\$1FFFFFF(~2.0meg)

BOARDS: None

Siehe auch:

CPU, VERSION

PAG Sandini

SKIP

Syntax:

2.0/2.1/3.0: SKIP [[LABEL] <sprungmarke>] [BACK]

Schablone:

2.0/2.1/3.0: LABEL, BACK/S

Pfad:

in die Shell integriert

Funktion:

Setzt die Ausführung einer Befehlsdatei an der mit <sprungmarke> bezeichneten Stelle fort.

Beschreibung: PAG Sandini

Der Befehl SKIP kann nur in Befehlsdateien eingesetzt werden. SKIP bewirkt das gleiche wie eine GOTO-Anweisung in Basic, eine Verzweigung zu einer bestimmten Stelle, wie sie in fast allen Hochsprachen zu finden ist. Die Stelle, an der die Abarbeitung der Befehlsdatei fortgeführt werden soll, muß mit dem Befehl LAB gekennzeichnet sein. Ausführliche Beispiele werden in den Beschreibungen zu den Befehlen IF und LAB gezeigt.

Es ist auch möglich SKIP keine Sprungmarke zu übergeben, er sucht dann nach einer Marke LAB, die ebenfalls keinen Namen besitzt. Dennoch sollte von dieser Möglichkeit kein Gebrauch gemacht werden:

SKIP

ECHO "SCH... Schule!"

LAB

Hier wird gar nichts ausgegeben.

Findet SKIP eine angegebene Sprungmarke nicht (oft aufgrund eines Tippfehlers), so springt SKIP an das Ende der Befehlsdatei und gibt folgende Fehlermeldung aus:

object not found
skip failed returncode 10

Argumente:

LABEL Der Name der Sprungmarke, die mit LAB <sprungmarke> gekennzeichnet sein muß. Die Bearbeitung der Befehlsdatei wird nach der LAB-Zeile fortgesetzt. Normalerweise sucht SKIP in einer Befehlsdatei nur vorwärts nach einer passenden LAB-Marke. Die Sprungmarke kann auch Leerzeichen enthalten, muß aber in diesem Fall in Anführungszeichen gesetzt werden. Andere Steuerzeichen (wie etwa Escape-Sequenzen) dürfen nicht verwendet werden. Der Befehl SKIP unterscheidet keine Groß- und Kleinbuchstaben, die Befehle "SKIP Ende" und "SKIP ENDE" sind gleichbedeutend, sie springen beide auf die erste der beiden Befehle "LAB ENDE" oder "LAB Ende".

BACK/S Es kann vorkommen, daß sich eine Sprungmarke in einer der darüber liegenden Zeilen befindet, so daß SKIP rückwärts nach der angegebenen Marke suchen soll. Dies wird SKIP mit diesem Schalter mitgeteilt. Es ist somit möglich, Schleifen zu programmieren. Wird SKIP bis zum Beginn der Befehlsdatei nicht fündig, beschwert der Befehl sich ebenfalls.

Siehe auch:

ENDKIP, EXECUTE, IF, LAB

SORT

Syntax:

2.0/2.1/3.0: SORT [FROM] <datei|muster> [TO] <datei>
[COLSTART <n>] [CASE] [NUMERIC]

Schablone:

2.0/2.1/3.0: FROM/A, TO/A, COLSTART/K/N, CASE/S, NUMERIC/S

Pfad:

C:

Funktion:

Sortiert die Zeilen einer Datei in alphabetischer Reihenfolge.

Beschreibung:

Mit SORT können die Zeilen der angegebenen Datei alphabetisch sortiert und in der TO-Datei abgelegt werden. Aber dieser Befehl hat nur sehr begrenzte Anwendungsmöglichkeiten - warum sollte ein Script oder ein anderer Text alphabetisch sortiert werden? Es gibt aber hin und wieder dennoch Situationen, in denen der Befehl eingesetzt werden kann, etwa um eine Adressenliste zu sortieren, deren einzelne Einträge in einer eigenen Zeile stehen und mit dem Nachnamen beginnen. Oder es soll die Ausgabe von LIST in alphabetischer Reihenfolge dargestellt werden, hierzu könnte folgendes Script eingesetzt werden:

```
.key verzeichnis
```

```
LIST >RAM:LISTOut$$
```

```
SORT RAM:LISTOut$$ TO RAM:SORTOut$$
```

```
TYPE RAM:SORTOut$$
```

```
DELETE >NIL: RAM:SORTOut$$
```

```
DELETE >NIL: RAM:LISTOut$$
```

Argumente:

FROM/A

Dieses Argument muß unbedingt angegeben werden und sagt SORT, welche Datei sortiert werden soll. Dies ist zwar im Normalfall eine Datei eines Datenträgers, es können aber auch andere Quellen wie etwa "PIPE:" oder das Shell-Fenster angegeben werden. Im

	letzteren Fall muß die Eingabe durch die Tasten CTRL+\ abgeschlossen werden.
TO/A	Hier wird der Name der Datei genannt, die die sortierten Zeilen aufnehmen soll. Auch hier kann es irgend ein beliebiges Ausgabegerät sein.
COLSTART/K/N	Da der Befehl SORT eine Datei zeilenweise sortiert, kann es von Zeit zu Zeit sein, daß das Kriterium, nach dem sortiert werden soll, nicht in der ersten Spalte beginnt. Ohne dieses Argument verwendet SORT immer die ersten Zeichen einer Zeile als Kriterium, an welcher Stelle die Zeile in der sortierten Datei eingesetzt wird. Es kann aber notwendig werden, daß die ersten Zeichen einer Zeile unbeachtet bleiben sollen und erst anschließend die Zeichen kommen, nach denen sortiert werden sollen. Mit dem Wert COLSTART kann SORT mitgeteilt werden, daß der Befehl die ersten <n> Zeichen beim Sortieren nicht berücksichtigen soll.
CASE/S	Normalerweise unterscheidet SORT keine Groß- und Kleinbuchstaben. Wird jedoch dieser Schalter gesetzt, so werden alle Kleinbuchstaben erst im Anschluß an den letzten Großbuchstaben eingesetzt. Anmerkung: In einigen Exemplaren der Workbench 2.0 wurde dieser Schalter falsch programmiert, daß er genau umgekehrt wirkt. Hier unterscheidet der Befehl voreinstellungsgemäß die Groß- und Kleinschreibweise, die Angabe dieses Arguments schaltet die Unterscheidung aus.
NUMERIC/S	Normalerweise werden Zahlen vor allen anderen Zeichen einsortiert, doch durch dieses Schlüsselwort werden die Buchstaben in der Reihenfolge noch vor den Zahlen eingesetzt. Dieses Schlüsselwort bewirkt aber noch etwas anderes: Die Zahlen werden als solche behandelt und nicht zeichenweise interpretiert. So werden Zahlen mit verschiedener Zeichenanzahl korrekt sortiert.

Beispiel:

In den beiden folgenden Fällen war die gleiche Datei, die nur Zahlen enthält, zu sortieren. Beim ersten Versuch wurden die Zahlen ziffernweise sortiert, beim zweiten wurden durch das Schlüsselwort NUMERIC die Werte der Zahlen berücksichtigt:

1. Work:DOS3.0> sort ram:test to *

0

1

10

1874

198

21

287

465

65

721

8

841

87

9

900

98

1. Work:DOS3.0> sort ram:test to * numeric

0

1

8

9

10

21

65

87

98

198

287

465

721

841

900

1874

PAG Sandini

Bemerkung:

Nach Commodores Angaben, können Probleme entstehen, wenn Dateien mit mehr als 200 Zeilen sortiert werden sollen, da der Stapelspeicher zu klein werden kann. Commodore empfiehlt, den Stapelspeicher mit dem Befehl **STACK** auf das 21-fache der Zeilenanzahl zu setzen. Nach dem Sortieren kann der Stapelspeicher wieder verkleinert werden.

Bei meinen geeigneten Tests konnte ich jedoch eine Datei mit über 3100 Zeilen mit dem voreingestellten Stapelspeicher von 4096 Bytes und sogar mit der minimal möglichen Größe von 1600 Bytes sortieren. Es ist daher zu vermuten, daß diese Meldung noch von älteren Versionen des Befehls **SORT** stammt, und der Befehl inzwischen überarbeitet wurde.

Siehe auch:

STACK

PAG Sandini

SOUND

Syntax:

- 2.0: Befehl nicht implementiert
- 2.1/3.0: SOUND [[FROM] <file>] [EDIT] [USE] [SAVE]
[PUBSCREEN <screen>]

Schablone:

- 2.0: Befehl nicht implementiert
- 2.1/3.0: FROM,EDIT/S,USE/S,SAVE/S,PUBSCREEN/K

Pfad:

SYS:Prefs

Funktion:

PAG Sandini

Es können die Einstellungen für den Alarm angezeigt oder verändert werden.

Beschreibung:

Wird SOUND ohne ein weiteres Argument oder mit dem Schlüsselwort "EDIT" aufgerufen, wird das Programm der Prefs-Schublade gestartet. Wird ein Dateiname angegeben und besitzt diese Datei Informationen zu den Voreinstellungen (sie wurde im SOUND-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert), können diese schon vorher definierten Einstellungen übernommen werden.

Argumente:

- FROM Name einer Datei, die im SOUND-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert wurde. Was mit diesen bereits definierten Einstellungen passiert, wird mit den anschließenden Schlüsselwörtern festgelegt.
- EDIT/S Startet das Programm "SOUND" der Prefs-Schublade, wie wenn das Icon auf der Workbench in der

Schublade Prefs angeklickt worden wäre. Dies ist auch dann der Fall, wenn überhaupt kein Argument übergeben wird.

USE/S

Wurde ein Dateiname einer SOUND-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen aktiviert. Beim Neustart des Rechners gelten jedoch wieder die alten Einstellungen.

SAVE/S

Wurde ein Dateiname einer SOUND-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen gespeichert und damit bei jedem Neustart des Rechners aktiviert.

PUBSCREEN/K

Wird hier ein Name eines anderen Screens eingegeben, so wird das SOUND-Fenster auf diesen Screen geöffnet.

PAG Sandini

SPAT

Syntax:

2.0/2.1/3.0: SPAT [COM] <command> [PAT] <pattern>
[<opt1>] [<opt2>] [<opt3>] [<opt4>]

Schablone:

2.0/2.1/3.0: COM/A,PAT/A,opt1,opt2,opt3,opt4

Pfad:

S:

Funktion:

Ermöglicht das sogenannten "Pattern matching" (die Verwendung von Jokerzeichen) auch für Programme, die von sich aus keine Jokerzeichen anerkennen.

Beschreibung: PAG Sandini

Dieser Befehl ist eigentlich gar keiner! Es handelt sich auch hier um ein weiteres Script, daß sich im Verzeichnis "S:" befindet. Unter Verwendung des AmigaDOS-Befehls LIST können hier andere AmigaDOS-Befehle, die einen Dateinamen als Parameter benötigen und normalerweise keine Jokerzeichen akzeptieren (wie etwa VERSION), indirekt mit Jokerzeichen arbeiten.

SPAT rechnet mit zwei übergebenen Argumenten, die anderen Optionen wurden noch eingebunden, falls ein über SPAT zu startender Befehl noch weitere Schalter oder Parameter benötigt.

Es sollte erwähnt werden, daß mit SPAT das aktuelle Verzeichnis nicht mit zwei Anführungszeichen "" und das Shell-Fenster nicht mit einem Stern * sondern mit zwei Sternen ** angesprochen werden kann.

Der Hintergrund dafür ist, daß SPAT seinerseits von dem Befehl EXECUTE ausgeführt wird, und dieser diese Zeichen anders interpretieren kann. Das aktuelle Verzeichnis kann ja über den Pfadnamen übergeben werden. Und wenn Sie die Ausgabe unbedingt in ein Konsolenfenster umleiten wollen, so sollten Sie dafür extra eines definieren "con:0/0...."

Argumente:

- COM/A Hier muß der Befehl eingegeben werden, der selbst keine Jokerzeichen kennt.
- PAT/A Hier wird das Namensmuster für die Dateien eingegeben, auf die der vorher eingetippte Befehl angewandt werden soll. Hier kann zwar auch ein Pfadname angegeben werden, doch die Jokerzeichen werden nur am Ende des Pfadnamens erwünscht, an der Stelle, an der normalerweise die konkreten Dateinamen stehen.
- opt1...4 Für den Fall, daß der eigentliche Befehl weitere Argumente benötigt, können bis zu vier weitere Parameter übergeben werden. Diese Argumente werden dem eigentlich auszuführenden AmigaDOS-Befehl direkt übergeben, sind also auch an den Befehl fest gebunden. Sollten in sehr seltenen Fällen diese vier Parameter nicht ausreichen, können mehrere Argumente durch umfassende Anführungszeichen zu einer Option für SPAT zusammengefügt werden. Auf den richtigen Befehl hat dies keinen Einfluß.

Bemerkung:

Da aber die meisten AmigaDOS-Befehle mittlerweile auch die Jokerzeichen kennen, wird SPAT nur noch sehr selten benötigt. Weil es aber auf Diskette nicht viel Platz beansprucht und es ab und an durchaus sinnvolle Anwendungen gibt, sollte diese Datei verschont bleiben - möglicherweise auch deswegen, weil der Anwender hier einige interessante Anregungen zu den Script-Dateien erhält. Schauen Sie sich doch einmal die Datei an.

Siehe auch:

DPAT

STACK

Syntax:

2.0/2.1/3.0: STACK [[SIZE] <n>]

Schablone:

2.0/2.1/3.0: SIZE/N

Pfad:

C:

Funktion:

Zeigt oder ändert die Größe des Stapelspeichers.

Beschreibung:

Jedes Programm - also auch die AmigaDOS-Befehle - benötigt zum Arbeiten einen sogenannten Stapelspeicher (engl: Stack). Es reicht jedoch dem normalen Anwender, wenn er weiß, daß der Befehl STACK zwei Funktionen haben kann. Er kann entweder die Größe des aktuellen Stacks zeigen, oder aber die Größe ändern. Normalerweise wird ein Anwender sehr selten die Größe des Stack ändern müssen, es sein den, er sortiert eine große Anzahl Daten (jedoch nicht mit dem Befehl SORT), er benutzt den ALL-Schalter bei den Befehlen DIR oder LIST, wenn die Tiefe der Verzeichnishierarchie mehr als sechs Stufen umfaßt oder benutzt den Befehl BRU.

Wird kein weiteres Argument übergeben, so zeigt STACK die aktuelle Größe des Stapelspeichers an:

```
1.Work:DOS3.0> stack  
Current stack size is 4096 bytes
```

Um die Größe zu ändern, wird einfach der Wert hinter dem Befehl angegeben:

```
1.Work:DOS3.0> stack 6000  
1.Work:DOS3.0> stack  
Current stack size is 6000 bytes
```

Hinweis: Wird von einer Shell aus ein neuer Prozess gestartet, so übernimmt dieser Prozess die Größe des Stapelspeichers von der startenden Shell. Programme müssen selbst überprüfen, ob der reservierte Stapelspeicher für ihre Aufgabe ausreicht und eine Warnung ausgeben, falls dies nicht der Fall ist. Der Amiga selbst zeigt keine Reaktion, wenn der Stack nicht mehr ausreicht – außer dem Guru.

Trotz dieser Tatsache ist es nicht ratsam, den Stapelspeicher jedesmal übermäßig zu dimensionieren, da in der Regel die eingestellte Größe von 4096 vollkommen ausreicht. Es wäre eine Verschwendung des kostbaren Speichers.

Argument:

SIZE/N Hier kann die neue Größe des Stapelspeichers eingegeben werden. Der niedrigste zulässige Wert ist 1600, aber die meisten Programme benötigen etwa 4000 bis 5000 Bytes. Es gibt aber auch Programme, die einen Stapelspeicher von 50000 und mehr fordern. In diesen Fällen geben sie jedoch einen entsprechenden Hinweis aus, damit der Anwender den geforderten Stack bereitstellen kann.

Siehe auch:

STATUS

STATUS

Syntax:

2.0/2.1/3.0: STATUS [[PROCESS] <n>] [FULL] [TCB] [CLI=ALL] [COMMAND <befehl>]

Schablone:

2.0/2.1/3.0: PROCESS/N, FULL/S, TCB/S, CLI=ALL/S, COM=COMMAND/K

Pfad:

C:

Funktion:

Zeigt Informationen über AmigaDOS-Prozesse.

Beschreibung: PAG Sandini

Mit diesem Befehl kann man sich Informationen beschaffen, wieviele und welche Prozesse des AmigaDOS gerade laufen, und unter welche Nebenbedingungen sie ihre Arbeit verrichten. Ein AmigaDOS-Prozess ist ein erweiterter Exec-Task, die Exec-Tasks können jedoch mit STATUS nicht angezeigt werden. Aber diese sind für AmigaDOS-Anwender sowieso eher unbedeutend (eben von niedrigerem Niveau).

Dennoch sollte der Unterschied erwähnt werden, da die AmigaDOS-Prozesse einige Privilegien gegenüber ihren Verwandten genießen, etwa eine Bevorzugung beim Zugriff auf die "dos.library".

Es gibt auch oft Prozesse, die existieren, aber gegenwärtig nichts zu tun haben. Diese werden oft leere Shell genannt, weil dies auch am meisten zutrifft - eine Shell wartet auf den nächsten Befehl. Wird der Befehl STATUS ohne weitere Argumente gestartet, so zeigt er die Liste aller laufenden und wartenden AmigaDOS-Prozesse.

Da nicht alle Prozesse ein eigenes Fenster besitzen - weil sie beispielsweise mit RUN gestartet wurden -, ist dies die einzige Möglichkeit, zu erfahren, ob ein derartiger Befehl noch arbeitet oder schon fertig ist. Das folgende Beispiel kommt von meinem Amiga 1200:

1.Work:DOS3.0> STATUS

Process 1: Loaded as command: STATUS

Process 2: Loaded as command: C:ConClip

Process 3: Loaded as command: hd0:m2emacs

Process 4: Loaded as command: sys:tools/commodities/blanker

Hier sieht man, daß der erste Prozess den Befehl STATUS abarbeitet, also für diese Ausgabe verantwortlich ist, der 2. ist mit CONCLIP beschäftigt, der dritte müht sich mit dem Editor M2Emacs ab und der vierte ist für den Bildschirmschoner "Blanker" zuständig.

Aber STATUS kann noch mehr Informationen ausgeben. Mit dem Schalter FULL werden so noch die Größe des Stapelspeichers (stk), die Größe der globalen Vektortabelle (siehe Bemerkungen) und die Priorität des Prozesses angezeigt werden:

1.Work:DOS3.0> STATUS FULL

Process 1: stk 4096, gv 150, pri 0 Loaded as command: STATUS

Process 2: stk 4000, gv 150, pri 0 Loaded as command: C:ConClip

Process 3: stk 4096, gv 150, pri 1 Loaded as command: hd0:m2emacs

Argumente:

PROCESS/N

Hier kann eine Prozessnummer eingegeben werden. Wird eine Nummer eingetippt, so zeigt STATUS nur die Informationen zu diesem einen Prozess, alle anderen werden ignoriert.

FULL/S

Mit diesem Schalter können weitere Informationen ausgegeben werden.

TCB/S

Dieser Schalter ist die Abkürzung für Task Control Block, wird er gesetzt, so erscheinen die Größen des Stacks und der Tabelle der globalen Vektoren, sowie die Priorität der Prozesse. Ob und welcher Befehl gerade abgearbeitet wird, bleibt dieses Mal geheim.

CLI=ALL/S

Dieser Schalter braucht nicht eingegeben werden, da der damit bezeichnete Modus bereits voreingestellt ist - es werden die Informationen zu allen AmigaDOS-Prozessen angezeigt.

COM=COMMAND/K

Hier kann ein Befehl genannt werden. STATUS sucht dann in der Prozessliste und gibt die Nummer des Prozesses aus, unter dem der genannte Befehl läuft. Ist derzeit kein derartiger Befehl in Arbeit, so wird ein Fehler erzeugt. Sind gleichzeitig mehrere Prozesse mit diesem Befehl beschäftigt, so wird nur die kleinste Prozessnummer ausgegeben. COMMAND muß dabei als Schlüsselwort angegeben werden. Diese Option kann in Verbindung mit dem Befehl BREAK sehr gut eingesetzt werden.

Bemerkungen:

Zu der merkwürdigen Tabelle der globalen Vektoren ist selbst fortgeschrittenen Amiga-Freaks wenig bekannt - sie weichen hier großzügig aus. Bemerkenswert ist, daß dieser Wert (gv) selbst sehr wenig aussagekräftig ist, da die Tabelle immer 150 Bytes groß ist. Der globale Vektor ist ein Relikt aus den Zeiten, in denen AmigaDOS noch mit der Programmiersprache BCPL geschrieben wurde. Hier sind ein paar Sprungadressen gespeichert, die von AmigaDOS-Programmen intern benutzt werden. Warum dieser Wert durch den Befehl STATUS angegeben wird ist weitgehend schleierhaft, vielleicht, weil die Tabelle eines der wenigen Dinge ist, die ein AmigaDOS-Prozess sein Eigen nennen kann, der größte Teil gehört Exec.

Siehe auch:

BREAK, CHANGETASKPRI, STACK

SUPER72

Syntax:

2.1/3.0: SUPER72 [HBSTRT=<n>] [HBSTOP=<n>] [HSSTRT=<n>]
[HSSTOP=<n>] [VBSTRT=<n>] [VBSTOP=<n>]
[VSSTRT=<n>] [VSSTOP=<n>] [MINROW=<n>]
[MINCOL=<n>] [TOTROWS=<n>] [TOTCLKS=<n>]
[BEAMCON0=<n>]

2.0: Befehl nicht implementiert

Schablone:

2.1/3.0: HBSTRT/K,HBSTOP/K,HSSTRT/K,HSSTOP/K,
VBSTRT/K,VBSTOP/K,VSSTRT/K,VSSTOP/K,MINROW/K,
MINCOL/K,TOTROWS/K,TOTCLKS/K,BEAMCON0/K

2.0: Befehl nicht implementiert

Pfad:

DEVS:Monitors

Funktion:

Bei diesem Befehl handelt es sich um einen weiteren Monitortreiber, der nur bei Verwendung eines Monitors mit einer Frequenz von 72 Hertz benutzt werden sollte.

Beschreibung:

Die Beschreibung des Befehls ADDMONTIOR gilt für SUPER72 sinngemäß, weshalb ich Sie auf den Befehl ADDMONITOR verweise - insbesondere beachten Sie bitte den Hinweis am Ende der Befehlsbeschreibung von ADDMONITOR.

Siehe auch:

ADDMONITOR

TIME

Syntax:

2.0: TIME [EDIT]
 2.1/3.0: TIME [EDIT] [SAVE] [PUBSCREEN <screen>]

Schablone:

2.0: EDIT/S
 2.1/3.0: EDIT/S, SAVE/S, PUBSCREEN/K

Pfad:

SYS:Prefs

Funktion:

Stellt die Systemuhr oder startet das gleichnamige Preferneces-Programm.

Beschreibung: PAG Sandini

Wird SCREENMODE ohne ein weiteres Argument oder mit dem Schlüsselwort "EDIT" aufgerufen, wird das Programm der Prefs-Schublade gestartet.

Argumente:

- EDIT/S Startet das Programm "TIME" der Prefs-Schublade, wie wenn das Icon auf der Workbench in der Schublade Prefs angeklickt worden wäre. Dies ist auch dann der Fall, wenn überhaupt kein Argument übergeben wird.
- SAVE/S Hiermit kann die aktuelle Zeit in die akkugepufferte Echtzeituhr gespeichert werden. Dies entspricht dem Befehl "SETCLOCK SAVE".
- PUBSCREEN/K Wird hier ein Argument übergeben, so wird das Fenster des Preferences-Programmes "TIME" in einem anderen Bildschirm als der Workbench geöffnet.

Siehe auch:

DATE, SETCLOCK, SETDATE

TYPE

Syntax:

2.0/2.1/3.0: TYPE [FROM] <{dateilmuster}> [TO <neuedatei>] [OPT
[IN] [HEX] [NUMBER]

Schablone:

2.0/2.1/3.0: FROM/A/M, TO/K, OPT/K, HEX/S, NUMBER/S

Pfad:

C:

Funktion:

Zeigt den Inhalt einer Textdatei oder einer Binärdatei.

Beschreibung:

Der Befehl TYPE ist wieder einer der wichtigsten und oft benutzten AmigaDOS-Befehle, der den Inhalt von Dateien ausgeben kann, ähnlich dem Programm More. Doch besitzt TYPE im Gegensatz zu MORE einige interessante Möglichkeiten, die Ausgabe zu gestalten oder gar das Ausgabemedium festzulegen. Die Ausgabe von sehr langen Dateien erfolgt ohne Pause zwischen den einzelnen Bildschirmseiten, es ist jedoch möglich, die Ausgabe durch drücken einer beliebigen Taste (mit einem druckbaren Zeichen) anzuhalten und mit der Löschtaste oder RETURN fortzufahren.

Der Befehl TYPE kann auch mehrere Dateien hintereinander ausgeben, indem entweder ein Namensmuster oder mehrere Namen der einzelnen Dateien als Argumente übergeben werden. Es ist oft auch angebracht, in Befehlsdateien längere Textpassagen nicht mit einer Unmenge von ECHO-Befehlen auszugeben, sondern eine Textdatei zu erstellen, die mit TYPE in einem Durchgang ausgegeben wird, so kann sehr viel Zeit gespart werden.

Argumente:

FROM/A/M

Hier muß mindestens ein Dateiname oder ein Namensmuster angegeben werden, es können aber auch mehrere sein. Normalerweise wird hier der Name einer normalen

Datei angegeben, wenngleich TYPE in der Lage ist, andere Quellen wie etwa PIPE: oder ein anderes Konsolen-Fenster zu benutzen. Wird eine andere Quelle benutzt, so muß die Eingabe mit dem EOF (End Of File)-Zeichen schließen, das mit den Tasten Ctrl+\ erzeugt wird.

TO/K

Normalerweise gibt TYPE den Inhalt der Dateien in das Shell-Fenster aus, in dem der Befehl eingegeben wurde. Es ist jedoch möglich, die Ausgabe in eine andere Datei oder ein anderes Gerät - beispielsweise den Drucker PRT:- umzuleiten. Hierzu dient dieses Schlüsselwort, es braucht nicht die Ausgabeumleitung nach der Technik ">ziel" eingesetzt werden. So ist es auch möglich, den Befehl TYPE zum Kopieren von Textdateien einzusetzen, bei anderen Dateien funktioniert dies jedoch nicht.

Werden mehrere Dateinamen oder Namensmuster angegeben, so kann TYPE auch die Funktion des Befehls JOIN übernehmen, indem diese Dateien in der TO-Datei aneinandergefügt sind.

OPT/K

Die Ausgabe von TYPE kann zusätzlich noch durch folgende zwei Optionen manipuliert werden:

HEX/S

(entspricht "OPT H"): Hierbei wird die Datei als sogenannter Hex-Dump ausgegeben. Hier ist die Anzeige unterteilt, die Spalte ganz links zeigt in einer vierstelligen Hexadezimalzahl die Anzahl der bereits dargestellten Bytes, es folgen vier Vierergruppen, in denen die einzelnen Bytes als zweistellige Hexadezimalzahl dargestellt werden. In der rechten Spalte werden diese Bytes als Zeichen interpretiert dargestellt. Besitzt ein Byte einen Wert, der nicht in ein druckbares Zeichen umgewandelt werden kann, erscheint an dessen Stelle ein Punkt ".". Siehe dazu die unten angegebenen Beispiele.

NUMBER/S

(entspricht "OPT N"): Hier werden die Zeilen der anzuzeigenden Textdatei am Zeilenbeginn durchnummeriert.

Beispiele:

Im ersten Fall habe ich einen Hex-Dump des Gadgets einer Schublade erstellt:

1.Work:DOS3.0> type hd1:TeX.info HEX

[illegible]

```

0210: AAAAAAAAA A8000555 55555555 50000000  TM TM TM TM ®..UUUUUP...
0220: 00000000 00003FFF FFFFFFFF F8003FFF  .....?.....-?.
0230: FFFFFFFF F8003FFF F600DFFF F8003FFF  ....-?.ö.B.-?.
0240: F62ADFFF F8003FFF F7FFDFFF F8003FFF  ö*ß.-?.~.B.-?.
0250: F8003FFF F8003FFF FFFFFFFF F8000000  -?..?.....-...
0260: 00000000 00000000 00000000 00000000  .....
0270: 00000000  ....

```

Sie können sich auch eine Befehlsdatei anzeigen lassen, in der die Zeilen durchnummeriert sind. Hierfür habe ich das Script SPAT ausgesucht:

```

1.Work:DOS3.0> TYPE S:SPAT OPT N
1.key com/a,pat/a,opt1,opt2,opt3,opt4
2.bra {
3.ket }
4
5;$VER: spat 38.1 (11.10.91)
6; Do wildcards for single arg commands
7
8 FailAt 21
9
10 List >T:q{$$} {pat} LFORMAT "{com} *"%s%s*" {opt1} {opt2} {opt3} {opt4}"
11
12 IF NOT FAIL
13 Execute T:q{$$}
14 Else
15 Echo "{pat} not found"
16 EndIF
17
18 FailAt 10

```

Siehe auch:

COPY, ECHO, JOIN

UNALIAS

Syntax:

2.0/2.1/3.0: UNALIAS [[NAME] <alias>]

Schablone:

2.0/2.1/3.0: NAME

Pfad:

in die Shell integriert

Funktion:

Löscht eine Definition, die mit dem Befehl ALIAS vereinbart wurde.

Beschreibung:

Dieser Befehl macht **genau das Gegenteil** des Befehls ALIAS - es entfernt ein Synonym aus der Alias-Liste einer Shell. Dieser Befehl wurde mit der Workbench 2.0 eingeführt, um die unter Workbench 1.3 verwendete Technik unnötig zu machen. Früher wurde eine Definition gelöscht, indem dem Befehl ALIAS lediglich die bereits vereinbarte Abkürzung ohne einen neuen Wert übergeben wurde.

Ohne irgend ein Argument zeigt UNALIAS genauso wie sein Gegenspieler die Liste der vereinbarten Synonyme. Weitere Informationen zu den Synonymen entnehmen Sie bitte der Beschreibung zu dem Befehl ALIAS.

Argument:

NAME Hier wird der Name des zu löschenden Synonyms eingegeben, es sollte eigentlich existieren, doch wird kein Fehler erzeugt, wenn die zu löschende Abkürzung nicht vereinbart ist.

Siehe auch:

ALIAS

UNSET

Syntax:

2.0/2.1/3.0: UNSET [[NAME] <lokaleVariable>]

Schablone:

2.0/2.1/3.0: NAME

Pfad:

in die Shell integriert

Funktion:

Löscht eine lokale Variable, die mit dem Befehl SET vereinbart wurde.

Beschreibung:

Dieser Befehl ist das Gegenstück zum Befehl SET. Er entfernt eine lokale Variable aus der Liste aller lokalen Umgebungsvariablen eines AmigaDOS-Prozesses. Wird UNSET ohne Argument benutzt, so zeigt der Befehl die Liste aller lokalen Umgebungsvariablen der aktuellen Shell. Mehr Informationen entnehmen Sie bitte der Beschreibung des Befehles SET.

Argument:

NAME Hier wird der Name der zu löschenden lokalen Umgebungsvariablen übergeben

Siehe auch:

SET, GET, SETENV, GETENV, UNSETENV

UNSETENV

Syntax:

2.0/2.1/3.0: UNSETENV [[NAME] <globale Variable>]

Schablone:

2.0/2.1/3.0: NAME

Pfad:

in die Shell integriert

Funktion:

Löscht eine globale Variable, die mit dem Befehl SETENV vereinbart wurde.

Beschreibung:

Dieser Befehl ist das Gegenstück zum Befehl SETENV. Er entfernt eine globale Variable aus der Liste aller globalen Umgebungsvariablen eines AmigaDOS-Prozesses. Wird UNSETENV ohne Argument benutzt, so zeigt der Befehl die Liste aller globalen Umgebungsvariablen der aktuellen Shell. Mehr Informationen entnehmen Sie bitte der Beschreibung des Befehles SETENV.

Argument:

NAME Hier wird der Name der zu löschenden globalen Umgebungsvariablen übergeben.

Siehe auch:

GET, GETENV, SET, SETENV, UNSET, UNSETENV

VERSION

Syntax:

2.0/2.1/3.0: VERSION [[NAME] <name>] [[VERSION] <version>]
 [[REVISION] <revision>] [[UNIT] <unitnr.>]
 [FILE] [INTERNAL | RES] [FULL]

Schablone:

2.0/2.1/3.0: NAME, VERSION/N, REVISION/N, UNIT/N,
 FILE/S, INTERNAL/S, RES/S, FULL/S

Pfad:

C:

Funktion:

Ermittelt die Versions- und Revisionsnummern einer Software.

Beschreibung:

Dieser Befehl wird in erster Linie dazu benutzt, um die Versionsnummer der benutzten Workbench und des Kickstarts zu ermitteln. Damit kann der Anwender jedoch sehr wenig anfangen. So kann man mit VERSION weiterhin auch die Versionsnummern von "Libraries" oder "Devices" ausgeben lassen, womit vorallem dem Programmierer schon wesentlich mehr geholfen ist. Es kommt ja auch oft vor, daß ein Anwender Teile des Betriebssystems erneuert jedoch andere dagegen unverändert läßt. Die gerade aktuelle Version, die ich während der Arbeit an diesem Buch verwende ist:

1. Work:DOS3.0> VERSION

Kickstart 39.106, Workbench 39.29

Wenn jedoch die Versionsnummer eines Device oder einer Library ermittelt werden soll, so braucht nur deren Name hinter dem Befehl VERSION angegeben werden:

> VERSION parallel.device

Es ist auch möglich, direkt zu testen, ob eine Library die erforderliche Versions- bzw. Revisionsnummer oder besser hat oder nicht. Zu diesem Zweck kann eine Zahl als Grenze angegeben werden. Wird diese Grenze durch die aktuelle Versionsnummer unterschritten, so wird der Fehlercode 5 (WARN) erzeugt, womit eine einfache Möglichkeit zum Test zur Verfügung steht. So kann man in Script-Dateien schon vor Programmaufruf prüfen, ob die erforderlichen Versionen vorhanden sind und einen entsprechenden aussagekräftigen Hinweis ausgeben, sofern dies nicht der Fall ist.

Generell zeigt VERSION die Nummern in der Form Version.Revision an, so bedeutet 39.109 beispielsweise Version 39, Revision 109. Werden dem Befehl VERSION jedoch Nummern übergeben, die er zu überprüfen hat, so müssen beide mit einem Leerzeichen voneinander getrennt werden. Wird nur die Revisionsnummer angegeben, so muß das Schlüsselwort REVISION unbedingt vorangestellt werden.

Neben der Workbench, Kickstart, den Devices und Libraries besitzen auch alle Befehle im Verzeichnis C: und viele weitere Programme eine Versionsnummer, die mit dem Befehl VERSION ermittelt werden kann. Dies ist gerade für Entwickler eine sehr große Hilfe, auch sollte der Anwender davon gebrauch machen, wenn er bei irgendeinem Befehl auf einen Fehler oder unsaubere Programmierung stößt. In diesem Fall kann er sich die Versions- und Revisionsnummer des Befehls merken und nach verbesserten Versionen ausschau halten. Oder besser: Commodore mitteilen, welcher Befehl in welcher Version welche Probleme bereitet. Da der Befehl VERSION keine Jokerzeichen akzeptiert, kommt hier das Makroprogramm SPAT zum Einsatz. Lassen Sie sich doch einmal die Versionsnummern aller Befehle im Verzeichnis C: zeigen:

```
1> SPAT Version C:#?
```

Argumente:

NAME

Hier kann der Name eines Befehls, Programmes, einer Library oder eines Devices angegeben werden, deren Versions- und Revisionsnummer ermittelt bzw. geprüft werden soll. Wird kein Name angegeben, so zeigt VERSION die Nummern für die Workbench und das Kickstart. Findet VERSION nichts, was zu dem angegebenen Namen paßt, so wird ein Fehler (Code 10) erzeugt.

- VERSION/N** Dieses Argument kann nur in Befehlsdateien eingesetzt werden, um zu prüfen, ob eine Software die erforderliche Versionsnummer (oder eine höhere) hat oder nicht. Ist die Versionsnummer niedriger als der hier angegebene Wert, so wird eine Warnung (Fehlercode 5) erzeugt, der mit der "IF WARN"-Anweisung abgefragt werden kann.
- REVISION/N** Dieses Argument kann nur in Befehlsdateien eingesetzt werden, um zu prüfen, ob eine Software die erforderliche Revisionsnummer (oder eine höhere) hat oder nicht. Ist die Revisionsnummer niedriger als der hier angegebene Wert, so wird eine Warnung (Fehlercode 5) erzeugt, der mit der "IF WARN"-Anweisung abgefragt werden kann. Dieses Argument sollte nur in Verbindung mit dem Argument VERSION benutzt werden.
- UNIT/N** In sehr seltenen Fällen besitzt das gleiche Device verschiedene Versionsnummern, wenn es verschiedene Geräte betreibt, etwa weil eine neuere Version mehr Geräte unterstützt, als die alte. So kann auch für ein bestimmtes Gerät die Versionsnummer eines Device ermittelt werden. UNIT sollte als Schlüsselwort genauso angegeben werden wie auch VERSION und REVISION.
- FILE/S** Wird dieser Schalter gesetzt, so wird bei einer Library, die auf der Diskette untergebracht ist, der komplette Pfad bei der Ausgabe von VERSION angegeben.
- INTERNAL/S** Ist ein Befehl oder eine Library bereits in das System eingebaut, so daß ein Laden von einer Diskette oder Festplatte nicht erforderlich ist, muß das Schlüsselwort INTERNAL angegeben werden. Bei einem Befehl muß dann nur noch der Befehlsname eingegeben werden. Wurde einem Befehl ein residenter Name zugewiesen, wo wird der ursprüngliche Name angezeigt.
- RES/S** Dieser Schalter ist eine Alternative zum Schlüsselwort INTERNAL. In späteren Versionen soll damit zwischen

resident gemachten Befehlen und solchen, die in die Shell integriert sind, unterschieden werden. Gegenwärtig sind jedoch RES und INTERNAL gleichbedeutend.

FULL/S

Mit diesem Schalter wird VERSION aufgefordert weitere Angaben zur Version zu machen. Zur Zeit wird oft das Datum der Kompilation in Klammern angezeigt:

```
1.Work:DOS3.0> version c:list full  
list 37.5 (08.11.91)
```

Siehe auch:

RESIDENT

PAG Sandini

WAIT

Syntax:

2.0/2.1/3.0: WAIT [<n>] [SEC | SECS] [MIN | MINS] [UNTIL <uhrzeit>]

Schablone:

2.0/2.1/3.0: /N, SEC=SECS/S, MIN=MINS/S, UNTIL/K

Pfad:

C:

Funktion:

Unterbricht die Arbeit für die angegebene Dauer.

Beschreibung:

Der Befehl WAIT ist offensichtlich wieder ein Vertreter der Familie der unnützen Befehle - warum sollte denn irgend ein Prozess seine Arbeit für eine Weile niederlegen, eine Gewerkschaft gibt es für Befehle nicht! Wer aber aufmerksam einige Befehlsdateien durchliest, der stößt immer wieder auf diesen Befehl. Der Befehl WAIT hält zwar einen Prozess auf, benötigt aber dafür keine Rechenzeit, so daß in der Zwischenzeit andere Prozesse oder Tasks ihre Arbeit verrichten können.

So wird der Befehl WAIT oft nach einem RUN-Befehl aufgerufen, der soeben einen Prozess gestartet hat, der sehr viel mit Disketten arbeitet. Damit kann ein unnötig häufiges Hin- und Herfahren des Schreib-/Lese-Kopfes vermieden werden, was den Nerven des Anwenders und auch dem Diskettenlaufwerk selbst gut tut. Obendrein kann dadurch Zeit gespart werden - so absurd dies im ersten Augenblick auch klingen mag. Denn dadurch, daß der eine Prozess alleine und vor allem ungestört arbeiten kann, brauchen beide Prozesse insgesamt weniger Zeit, als wenn beide gleichzeitig versuchen, ihre Aufgaben zu erfüllen und sich somit gegenseitig behindern.

Wird WAIT ohne ein Argument angegeben, so wartet WAIT genau eine Sekunde lang.

Argumente:

- /N** Hier wird zunächst nur eine Zahl angegeben. Wie WAIT diese Zahl zu interpretieren hat, sagen erst die folgenden Schlüsselwörter.
- SEC=SECS/S** Wird dieses Schlüsselwort angegeben, so wird die Zahl in Sekunden interpretiert, WAIT wartet n Sekunden. SEC oder SECS schließt natürlich das Schlüsselwort MIN bzw. MINS aus, es kann immer eines von beiden angegeben werden.
- MIN=MINS/S** Mit diesem Schalter wird die angegebene Zahl in Minuten interpretiert, WAIT wartet n Minuten. Dieses Schlüsselwort kann nicht gleichzeitig mit SEC oder SECS verwendet werden.
- UNTIL/K** Man kann WAIT auch explizit eine Uhrzeit angeben, bis zu der der Prozess angehalten werden soll. Diese Zeit kann nur in Stunden und Minuten angegeben werden. Anmerkung: Der Befehl WAIT arbeitet unter Verwendung des Arguments UNTIL nicht sehr genau, der jeweilige Prozess wird auch nach Erreichen der eingestellten Zeit noch etwas angehalten. Dies hängt auch davon ab, welche Priorität der entsprechende Prozess im Verhältnis zu seinen Mitbewerbern (um die Rechenzeit) besitzt. Je niedriger diese Priorität ist, desto länger wird die Verzögerung werden.

Siehe auch:

BREAK, RUN

WBPATTERN

Syntax:

2.0: WBPATTERN [[FROM] <file>] [EDIT] [USE] [SAVE]
[WORKBENCH] [WINDOWS]

2.1/3.0: WBPATTERN [[FROM] <file>] [EDIT] [USE] [SAVE]
[CLIPUNIT <n>] [NOREMAP]

Schablone:

2.0: FROM, EDIT/S, USE/S, SAVE/S, WORKBENCH/S, WINDO-
WS/S

2.1/3.0: FROM, EDIT/S, USE/S, SAVE/S, CLIPUNIT/K/N, NOREMAP/S

Pfad:

SYS:Prefs

PAG Sandini

Funktion:

Es können die Einstellungen für die Hintergrundmuster der Workbench- oder Schubladen-Fenster angezeigt oder verändert werden.

Beschreibung:

Wird WBPATTERN ohne ein weiteres Argument oder mit dem Schlüsselwort "EDIT" aufgerufen, wird das Programm der Prefs-Schublade gestartet. Wird ein Dateiname angegeben und besitzt diese Datei Informationen zu den Voreinstellungen (sie wurde im WBPATTERN-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert), können diese schon vorher definierten Einstellungen übernommen werden.

Argumente:

FROM

Name einer Datei, die im WBPATTERN-Programm mit dem Menüpunkt "Save As" oder "Speichern als" abgespeichert wurde. Was mit diesen bereits definierten Einstellungen passiert, wird mit den anschließenden Schlüsselwörtern festgelegt.

EDIT/S	Startet das Programm "WBPATTERN" der Prefs-Schublade, wie wenn das Icon auf der Workbench in der Schublade Prefs angeklickt worden wäre. Dies ist auch dann der Fall, wenn überhaupt kein Argument übergeben wird.
USE/S	Wurde ein Dateiname einer WBPATTERN-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen aktiviert. Beim Neustart des Rechners gelten jedoch wieder die alten Einstellungen.
SAVE/S	Wurde ein Dateiname einer WBPATTERN-Einstellung genannt, so werden diese bereits vorgenommenen Einstellungen gespeichert und damit bei jedem Neustart des Rechners aktiviert.
WORKBENCH/S	Hier wird nur das Muster für den Workbench-Hintergrund aktiviert. Wird dieses Schlüsselwort in Verbindung mit SAVE verwendet, so wird diese Einstellung ebenfalls gespeichert.
WINDOWS/S	Hier wird nur das Hintergrundmuster für die Schubladen-Fenster verwendet, die Workbench bleibt davon unberührt. Wird dieses Schlüsselwort in Verbindung mit SAVE verwendet, so wird diese Einstellung ebenfalls gespeichert. Dieser Schalter sollte jedoch in der gegenwärtigen Version von WBPATTERN vorsichtig eingesetzt werden, da ungeliebte außergewöhnliche Nebenerscheinungen zu Tage treten können.

WHICH

Syntax:

2.0/2.1/3.0: WHICH [FILE] <befehl> [NORES] [RES] [ALL]

Schablone:

2.0/2.1/3.0: FILE/A, NORES/S, RES/S, ALL/S

Pfad:

C:

Funktion:

Zeigt, in welchem Verzeichnis der angegebene Befehl zu finden ist.

Beschreibung:

Mit diesem Befehl kann der Pfad einer beliebigen Datei gefunden werden, wenn sich die Datei im aktuellen Suchpfad befindet. WHICH ist dem Befehl SEARCH sehr nah verwandt, aber unvorstellbar langsamer. Normalerweise durchsucht WHICH zuerst die Liste der residenten Befehle, dann das aktuelle Verzeichnis und anschließend die anderen Verzeichnisse entlang des Suchpfades. Wird WHICH nirgends fündig, gibt der Befehl die Suche auf.

Findet der Befehl auf seiner Suche jedoch mehrere passende Dateien, so werden alle möglichen Pfade angegeben.

Argumente:

FILE/A Natürlich benötigt WHICH den Namen der Datei, die er suchen soll. Jokerzeichen sind hier nicht zugelassen. Sollte es sich um einen residenten Befehl handeln, so sucht WHICH nach dem residenten Namen, der sich von dem der Diskette unterscheiden kann.

NORES/S Wird dieser Schalter gesetzt, so wird die Liste der residenten Befehle nicht durchsucht.

- RES/S** Dieser Schalter bewirkt genau das Gegenteil - hier sucht WHICH ausschließlich in der Liste der residenten Befehle nach der angegebenen Datei. Es ist nicht erforderlich, eine Datei, die kein ausführbares Programm darstellt, unter den residenten Befehlen zu suchen, diese Datei dort gar nichts zu suchen hat.
- ALL/S** Hiermit kann WHICH aufgefordert werden, alle Pfade zu zeigen, in denen eine Datei mit dem angegebenen Namen vorhanden ist.

Beispiele:

Sie suchen den Befehl CD? Kein Problem - er ist bereits in die Shell integriert:

```
1.Work:DOS3.0> which CD
INTERNAL CD
```

Wo steckt den der Befehl Search?

```
1.Work:DOS3.0> which search
Workbench:C/Search
```

Bei den anderen AmigaDOS-Befehlen im Verzeichnis C:!

Und CMD...

```
1.Work:DOS3.0> which cmd
Workbench:Tools/CMD
```

... findet man bei den Tools wieder.

Siehe auch:

PATH, RESIDENT

WHY

Syntax:

2.0/2.1/3.0: WHY

Schablone:

2.0/2.1/3.0: -

Pfad:

in die Shell integriert

Funktion:

Erzeugt ein vorangegangener Befehl einen Fehler, so kann mit dem Befehl WHY eine detailliertere Beschreibung angefordert werden.

Beschreibung:

Was geht Ihnen als erstes durch den Kopf, wenn ein Befehl versagt? Pssst - und was als zweites? Richtig: Warum? Oder als echter Computerfreak: WHY? Genau das macht der Befehl WHY - er gibt detailliertere Informationen, warum der vorangegangene Befehl nicht korrekt arbeiten konnte.

Die meisten Programme geben aber schon von sich aus sehr genaue Erklärungen, aus welchem Grund die eine oder andere Aktion nicht durchgeführt werden konnte. Der Befehl WHY scheint also weitgehend überflüssig zu sein. Es gibt aber auch Situationen, in den sich die Befehle sehr wortkarg geben ("bad args"), "<befehl> failed returncode ##") oder gar in tiefes Schweigen hüllen. Dieses Schweigen tritt vor allem dann auf, wenn die Ausgabe ins Nichts umgeleitet wurde. So kann man schließlich überprüfen, ob der Befehl korrekt ausgeführt werden konnte oder nicht. Die Anwendung von WHY ist denkbar einfach - Sie warten, bis ein Befehl nicht korrekt gearbeitet hat und geben dann WHY ein. Aber machen Sie sich keine Illusionen, die Meldungen von WHY sind oft ebenfalls nicht sehr aussagekräftig.

WHY kann nur die Rückgabewerte des unmittelbar vorher ausgeführten Befehles untersuchen - genaugenommen wird der Inhalt der lokalen Varia-

ben "Result2" interpretiert. Wird hier nur der Wert Null aufgefunden, zuckt WHY mit der Schulter: "the last command did not set a return code" (frei übersetzt: "Ich weiß nicht, was passiert ist!")

In der Praxis macht WHY nichts anderes, als daß er den Wert der lokalen Umgebungsvariablen "Result2" dem Befehl FAULT übergibt und die Rückmeldung ausgibt. Fehlt der Befehl FAULT, hat auch WHY nichts zu melden. Wohl auch aus diesem Grund sind beide Befehle bereits in die Shell integriert.

Siehe auch:

FAULT, GET, SET

PAG Sandini

;

Syntax:

2.0/2.1/3.0: ; Kommentar

Schablone:

2.0/2.1/3.0: -

Pfad:

in AmigaDOS integriert

Funktion:

Leitet den Kommentar ein.

Beschreibung:

Dieser Befehl ist eigentlich gar keiner. Mit dem Semikolon “;” wird für den Parser (der Programmteil der Shell oder von EXECUTE, der die Befehlszeile analysiert) angezeigt, daß alle folgenden Zeichen in der Zeile für den Befehl keine Bedeutung mehr haben und lediglich als Kommentar angefügt wurden.

Der Kommentar hat überhaupt keinen Einfluß auf den Befehl, er dient lediglich für den Anwender, um beispielsweise Befehlsdateien verständlicher zu machen. AmigaDOS ignoriert beim Auftreten eines Semikolons alle folgenden Zeichen in dieser Zeile.

<, >, >>

Syntax:

2.0/2.1/3.0: <AmigaDOS-Befehl> >ziel oder
<AmigaDOS-Befehl> >>ziel oder
<AmigaDOS-Befehl> <quelle

Pfad:

in die Shell integriert.

Funktion:

Leitet die Ausgabe in eine Datei oder ein Gerät um oder holt die Eingabe von einer Datei bzw. einem Gerät.

Beschreibung:

Diese spitzen Klammern sind eigentlich kein Befehl, sie werden als Operatoren bezeichnet. Diese Operatoren werden sehr oft benutzt, um Ein- bzw. Ausgaben von bzw. zu einem Gerät umzuleiten. Bitte verwechseln Sie aber diese Operatoren nicht mit den spitzen Klammern in der Beschreibung der Syntax der AmigaDOS-Befehle, da diese lediglich kennzeichnen, daß der Anwender an der entsprechenden Stelle irgendwelche Daten nach seinen Vorstellungen eingeben muß. Zur Unterscheidung: Bei der Befehlsbeschreibung tragen die spitzen Klammern jeweils vor und nach dem Wert auf, in der Eingabe des Befehls sind diese und alles was dazwischen steht durch einen Wert, den der Anwender festlegt, zu ersetzen. In allen obigen Beschreibungen der Befehlssyntax kommt keine einzige Eingabe- oder Ausgabeumleitung vor!

Die Wirkungsweise der einzelnen Operatoren:

- > Die wohl am häufigsten verwendete Umleitung von Daten geschieht bei der Ausgabe, sie ist auch am einfachsten zu handhaben. Jedesmal wenn ein ">ziel" zwischen dem Befehlsnamen und dem ersten Argument steht wird der Text, der normalerweise im Shell-Fenster erscheint, in der Datei "ziel" gespeichert. Natürlich kann für "ziel" jeder beliebige Dateiname oder komplette Pfade - auch

mit logischen Geräten beginnend - eingesetzt werden. Achtung: Wird die Ausgabe mit diesem Operator in eine Datei umgeleitet, die bereits vorhanden war, so werden alle vorherigen Daten dieser Datei gelöscht. Soll der Text jedoch an die bereits bestehende Datei angefügt werden, so muß eine Ausgabeumleitung mit ">>ziel" (siehe dort) durchgeführt werden.

< Mit diesem Operator kann einem Befehl gesagt werden, er solle sich alle weiteren benötigten Daten nicht von der Shell sondern von der angegebenen Quelle besorgen. Dies hört sich genau so leicht an, wie die Ausgabeumleitung, dem ist aber bei weitem nicht so. Da ein bestimmter Befehl in der Regel ganz bestimmte Daten in einer festgelegten Reihenfolge benötigt, muß die Quelldatei (es kann auch eine andere Quelle sein wie etwa ein eigenes Konsolenfenster oder die serielle Schnittstelle "AUX:") diesen Anforderungen genügen. Es kann nicht jede beliebige Quelle verwendet werden, der Anwender muß darauf achten, daß der Befehl vernünftige Daten bekommt. Ist dies nicht der Fall, arbeitet der Befehl nicht vernünftig.

In der Praxis zeigt sich aber, daß der Durchschnittsanwender die Eingabeumleitung nicht benötigt. Trotzdem darf diese Möglichkeit nicht unterschätzt werden. So können beispielsweise Programmierer beim Testen eines Programmes sicherstellen, daß ein Programm bei mehreren Testläufen immer identische Daten bekommt. Aber Anwender, die sich so intensiv mit dem Amiga beschäftigen, besitzen ja ausreichend Erfahrung, daß sie die Anforderungen, die an die Quelldatei gestellt werden, kennen und erfüllen werden.

>> Auch dieser Operator stellt eine Ausgabeumleitung dar, er wird jedoch nur sehr selten eingesetzt. Die Auswirkungen sind im wesentlichen identisch mit dem Operator ">", der wichtige Unterschied ist der, daß eine bereits bestehende Zieldatei nicht gelöscht wird, wenn die Ausgabe mit ">>" dort hingeleitet wird, sondern der Text, der normalerweise im Shell-Fenster erscheint wird an die bereits bestehende Datei angefügt. Es wird jedoch gefordert, daß die Zieldatei bereits besteht, andernfalls wird eine

Fehlermeldung ausgegeben. Es wird jedoch nicht gefordert, daß die Zieldatei bereits Daten enthalten muß, sie kann also auch leer sein. Sie können also die Zieldatei vorher mit dem Befehl

```
1> ECHO >T:Ziel "" NOLINE
```

erstellen und anschließend immer wieder mit >>T:Ziel Text anfügen.

Bemerkung:

In der offiziellen Dokumentation zu den AmigaDOS-Befehlen der Workbench 3.0 steht, daß die Ausgabe- bzw. Eingabe-Umleitung direkt nach dem Namen des entsprechenden Befehls und vor allen anderen Argumenten stehen muß. Bei meinen Test mit den Befehlen DIR, LIST und COPY hat sich jedoch ergeben, daß die Ausgabeumleitung auch an anderer Stelle stehen darf und trotzdem akzeptiert wird. So kann man zwar sagen, daß die Datenumleitung nicht unbedingt direkt hinter dem Befehlsnamen stehen muß, aber dennoch dort stehen sollte. Damit kann von Anfang an Mißverständnissen beim Lesen von Befehlen (etwa in Script-Dateien) vorgebeugt werden.

In Befehlsdateien kann auch noch ein weiterer Punkt falsch interpretiert werden: Die Klammern in Texten, mit denen sich einzelne Befehle auf die Argumente eines Scripts beziehen. Einer Befehlsdatei können ja beim Aufruf Argumente übergeben werden (siehe dazu auch EXECUTE), auf diese Werte wird dann innerhalb der Befehlsdatei zugegriffen, in dem der Name eingeschlossen in zwei spitze Klammern geschrieben wird. Um hier eine Verwechslung mit der Datenumleitung auszuschließen, empfiehlt es sich, die Kennzeichnung der Argumente durch die .BRA- und .KET-Anweisung (wird bei EXECUTE beschrieben) zu ändern.

3.7 MEMACS - Der Texteditor

Bevor wir uns nocheinmal etwas ausführlicher den Befehlsdateien oder Scripts zuwenden, sollten wir uns mit einem Werkzeug vertraut machen, damit die Befehlsdateien auch bearbeitet werden können. Das wichtigste Hilfsmittel ist ein Texteditor, der Text als reine ASCII lesen und schreiben kann. Beim Amiga werden auf den Systemdisketten gleich drei Editoren mitgeliefert: "ED", "EDIT" und "MEMacs". EDIT ist für diesen Zweck denkbar ungeeignet, und ED bietet zwar ausreichend Möglichkeiten zur Textmanipulation, ist aber nicht so komfortabel und umfangreich wie sein schärfster Konkurrent MEMacs. Bei letzterem fehlt eigentlich nur noch ein richtiger Dateiauswahlrequester (den es ja auch schon vorbereitet gäbe!), dann würde ich keine weiteren Wünsche mehr an ihn stellen.

MEMcas ist ein komfortabler Bildschirm-Editor, der mit einem umfangreichen Menü versehen ist. Alle Menüpunkte können auch als Tasten-Kombinationen (sogenannte "Short Cuts") aufgerufen werden, so daß ein geübter Anwender sehr schnell alle wichtigen Funktionen über die Tastatur aufrufen kann und die Maus gar nicht mehr benötigt. Wer nun denkt, "was hat er nur gegen die Maus?", der sollte sich einmal überlegen, wie mühevoll es ist, wenn er wegen jedem einzelnen Menüpunkt wieder die Finger von der Tastatur nehmen soll (schließlich will man ja einen Text eingeben) um mit der Maus die gewünschte Funktion im Menü auswählen muß. Je wichtiger und häufiger benutzt ein Menüpunkt wird, desto schneller wird man sich auch die entsprechende Abkürzung einprägen können.

Es ist sogar möglich, mit MEMacs gleichzeitig mehrere verschiedene Texte zu bearbeiten, allerdings nur, wenn alle diese Texte gleichzeitig in den Speicher passen. Die Breite des darstellbaren Textes ist durch die Breite des Bildschirms festgelegt, normalerweise sind dies 80 Zeichen. Unter normalen Bedingungen reicht dies vollkommen aus und falls nicht, gibt es Tricks, um auch längere Zeilen eingeben bzw. bearbeiten zu können. Wird eine Zeile zu lang, so werden die hinausragenden Zeichen nicht mehr angezeigt. Um dem Anwender dennoch mitzuteilen, daß die Zeile eigentlich länger ist, aber nicht vollständig dargestellt wird, zeigt MEMacs dann an der rechten Grenze des dargestellten Textes ein Dollarzeichen \$. Wenn Sie mit dem Cursor auf das Zeichen unmittelbar links des Dollarzeichens fahren und die Return-Taste

drücken, erscheinen in der folgenden Zeile die nächsten 80 Zeichen (allerdings als eigene Zeile). Wollen sie die Zeile wieder im ursprünglichen Zustand herstellen, so muß der Cursor auf des erste Zeichen der unteren Zeile gefahren werden und einmal (!) die Backspace (Löschtaste) gedrückt werden. So wird die untere Zeile an die obere angefügt, es erscheint wieder das Dollarzeichen.

Sollte auch die neue Zeile nicht komplett dargestellt werden können so kann das oben geschilderte Verfahren wiederholt werden, bis die gesamte ursprüngliche Zeile in einzelne Zeilen aufgeteilt darstellbar ist. Umgekehrt können auch die einzelnen Teile wieder zu einem Ganzen zusammengefügt werden.

Einen kleinen Nachteil gibt es aber schon noch, der nicht unerwähnt bleiben sollte: MEMacs kennt zwar die deutschen Umlaute ä, ö, ü und ß, behandelt diese aber nicht als Buchstaben, sondern wie Satzzeichen. Das ist zwar nicht weiter schlimm, solange man keine Befehle verwendet, die auf Wörter angewandt werden. Bezieht sich eine Funktion auf Wörter (beispielsweise "Mache ein Wort groß"), so endet für MEMacs ein Wort bei einem Umlaut, auch wenn dieser mitten drin steht.

3.7.1 *Der Aufruf von MEMacs*

Der Aufruf des Programm kann von der Workbench aus über das zweimalige Anklicken des entsprechenden Symboles in der Schublade "Tools" der Diskette "ExtrasDisk" erfolgen oder von der Shell aus durch Eingabe des Befehls MEMACS. Welche Argumente dabei benutzt werden können, sehen Sie in der Beschreibung des Befehls MEMACS.

Wenn Sie den Editor MEMcas aufrufen, ohne den Namen der zu bearbeiten den Datei zu übergeben, so erscheint am unteren Bildschirm - bzw. Fensterrand die Zeile "`— MicroEMACS — main —`", darunter die Zeile "`[New file]`".

Wie oben schon angedeutet wurde, kann MEMacs gleichzeitig auch mehrere Dateien bearbeiten. Dazu müssen mehrere Textspeicher, sogenannte "Buf-

fer" verwaltet werden. Um die einzelnen Textspeicher voneinander zu unterscheiden, erhalten Sie einen Namen. Wird in den Textspeicher eine Datei geladen, so erhält der Textspeicher den gleichen Namen wie die Quelldatei, wurde keine Datei eingelesen, so wird der Buffer "main" (Hauptpuffer) genannt. MEmacs besitzt immer mindestens einen Textspeicher. In der Anzeige von MEmacs wird in der vorletzten Zeile der Name des Textspeichers (beispielsweise "main") unmittelbar rechts neben dem Programmnamen MicroEmacs und der Name der Datei (nicht vorhanden, wenn keine Datei geladen wurde) wiederum rechts davon angezeigt. Ob der Originaltext seit dem Laden in MicroEmacs bereits verändert wurde, erkennen Sie an dem Stern "*", der unmittelbar links neben dem Wort "MicroEmacs" auftaucht, wenn ein Zeichen eingegeben wird.

Wenn Sie gleichzeitig mit mehreren Textspeicher arbeiten, können Sie zu jeder Zeit den gewünschten Buffer auf dem Bildschirm darstellen lassen. Das Umschalten zwischen den einzelnen Textspeicher erfolgt über das Menü. Der jeweils angezeigte Text entspricht dem aktuellen Inhalt des Textspeichers.

MicroEmacs unterscheidet zwei verschiedene Modi, einen normalen Modus, in dem der Text direkt bearbeitet werden kann, und einen sogenannten Befehlsmodus, auf letzteren werden wir noch häufiger stoßen. Im normalen Modus sind folgende Funktionen möglich:

- * An der aktuellen Cursorposition können Zeichen über die Tastatur in den Text eingefügt werden.
- * Mit der Taste "Del" wird das Zeichen unter dem Cursor gelöscht.
- * Mit der Lösch taste ("Backspace") wird ein Zeichen unmittelbar links vom Cursor gelöscht und der Cursor ein Zeichen nach links versetzt.
- * Der Cursor kann mit den Cursortasten beliebig im Text platziert werden.
- * In Kombination mit der Shift-Taste kann der Cursor mit den Cursortasten an den Rand des Fensters gebracht werden.
- * Durch anklicken einer beliebigen Stelle im Text mit der Maus kann der Cursor dort gesetzt werden.
- * Jeder beliebige Menüpunkt kann ausgewählt werden.
- * Weitere Möglichkeiten werden im Folgenden erläutert.

Der Befehlsmodus unterscheidet sich vom normalen Modus darin, daß der Cursor in die letzte Zeile des Bildschirms springt, und das Programm den Anwender auffordert, dort bestimmte Informationen einzugeben. Der Befehlsmodus wird über verschiedene Menüpunkte aktiviert. Um die späteren Erläuterungen besser zu verstehen, sollten Sie sich folgende Begriffe einprägen, die immer wieder benutzt werden:

Buffer Ein Buffer ist ein Textspeicher, in dem immer nur eine Datei also immer nur ein Text geladen und bearbeitet werden kann. Für jede weitere Datei kann eigener Buffer eingerichtet werden.

.(“Dot”) Aktuelle Position des Cursors im aktuellen Textspeicher.

Mark Es ist oft notwendig, einen ganzen Textbereich zu bearbeiten, hierzu muß der Beginn dieses Bereiches über das Menü oder eine entsprechende Tastenkombination gekennzeichnet werden. Diese Stelle wird “Mark” genannt. Jeder Textspeicher hat seine eigene Markierung. Eine so gekennzeichnete Position wird solange beibehalten, bis in dem jeweiligen Textspeicher eine neue Position markiert wird. Leider kann MEMacs diese Markierungen nicht optisch kennzeichnen. Wenn später von einer Marke die Rede ist, so ist immer die letzte Stelle gemeint, die über das Menü oder die entsprechende Tastenkombination gekennzeichnet wurde.

Window Wenn Sie mit MEMcas arbeiten und dort ein weiteres Fenster öffnen wollen, werden Sie nicht das sehen, was Sie sich wohl unter einem Fenster vorstellen. Ein Fenster von MEMacs unterscheidet sich von einem normalen Fenster der Workbench (eigentlich von einem Fenster der “intuition.library”) in vielen Punkten. Sie finden weder Rahmen noch Gadgets, es wird lediglich der normale Bildschirm geteilt, die ursprünglich nur vorletzte Zeile wird in der Mitte des Bildschirms noch einmal dargestellt. Was Sie jetzt sehen sind zwei MEMacs-Fenster, in der oberen und in der unteren Hälfte des Bildschirms je eines (mit möglicherweise ganz anderem Inhalt). Der Bildschirm wird also horizontal in mehrere Bereiche unterteilt, jeder Bereich bei MEMacs als Fenster bezeichnet.

Nach sovielen Definitionen wollen wir uns nun das Menü des Editors zu Gemüte führen.

3.7.2 Das Menü des MEmacs

In diesem Abschnitt wird das Menü und dessen Funktionen detailliert beschrieben. Das Menü ist ausschlaggebend für die Qualität eines Editors, MEmacs stellt sehr viele notwendige Funktionen zur Verfügung, jedoch nur die wirklich wichtigen, unnötige Punkte wurden weggelassen. Somit ist ein sehr kompaktes Menü entstanden, das sehr schnell beherrscht werden kann. Der Editor MEmcas stellt folgende Menüs zur Auswahl:

- Project** Unter diesem Titel finden Sie Funktionen, die sich auf Dateien, Textspeicher und ähnliches beziehen.
- Edit** Dieses Menü enthält Funktionen zum Manipulieren von Textbereichen.
- Window** Mit diesem Menü können Aktionen in Bezug auf die Fenster durchgeführt werden.
- Move** Dieses Menü ermöglicht schnelle Cursor-Bewegungen.
- Line** Mit diesem Menü werden zeilenbezogene Aktionen aufgerufen.
- Word** Wie der Name schon vermuten läßt, können hiermit Wörter manipuliert werden.
- Search** Dieser Titel beherbergt umfangreiche Such- und Ersetz-Routinen.
- Extras** Hier werden in erster Linie sogenannte Macro-Befehle verwaltet.

Zu jedem einzelnen Menüpunkt gibt es eine bestimmte Tastenkombination, die die gleiche Aktion auslösen kann. Die äquivalente Tastenkombination - im englischen "Short Cut" genannt - steht auch hinter dem jeweiligen Menüpunkt, so daß jederzeit eine Kombination wieder gezeigt werden kann, wenn sie eventuell vergessen wurde.

Jede Tastenkombination wird entweder mit der Escape-Taste "Esc" eingeleitet oder man muß die Ctrl-Taste gleichzeitig mit einer bestimmten anderen Taste drücken, in wenigen Fällen werden beide Tasten verlangt. Dabei ist an-

zumerken, daß die Esc-Taste immer alleine und unmittelbar vor der nachfolgenden Taste zu drücken ist, die Ctrl-Taste jedoch gleichzeitig mit der angegebenen Taste. Werden beide gleichzeitig genannt, so ist erst die Esc-Taste alleine zu drücken und anschließend die Ctrl-Taste gleichzeitig mit der weiteren.

Im Menü erscheinen die Buchstaben "ESC", wenn die Escape-Taste vorausgeschickt werden soll und das Zeichen "^", wenn die Ctrl-Taste gleichzeitig zu drücken ist. Es ist auch möglich, daß hinter der Ctrl-Taste und dem ersten ein weiteres Zeichen eingegeben werden muß. Ist dabei erneut die Ctrl-Taste gleichzeitig zu drücken, wird auch diesem Zeichen das "^" vorangestellt. Bei Sonderzeichen, wie etwa dem At-Zeichen "@" muß auch noch die Alt-Taste gleichzeitig gedrückt werden.

Nach sovielen Tasten ist aber jetzt die Zeit für das Menü gekommen. Bitte sehr:

3.7.2.1 Das Menü "Project"

Die meisten der hier zu findenden Menüpunkte beziehen sich auf den aktuellen Textspeicher, in dem sich der Cursor befindet.

Rename Ctrl-X, F

Hier wird der Name des aktuellen Textspeichers geändert, der Name der Quelldatei auf dem Datenträger bleibt dabei unverändert. Dabei wird der Befehlsmodus des MEMacs aktiviert, in der letzten Zeile erscheint die Eingabeaufforderung "New file name:". An dieser Stelle werden Sie gebeten, den neuen Namen der Datei einzugeben. Eigentlich ist diese Eingabeaufforderung etwas irreführend, da der Name der Datei unverändert bleibt. Wenn jedoch der Text mit dem Menüpunkt "Save - file" (siehe weiter unten) wieder abgespeichert wird, so wird eine neue Datei mit dem entsprechenden neuen Namen erstellt, die ursprüngliche Textdatei bleibt unverändert. Wurde jedoch kein neuer Name eingegeben und stattdessen nur die Return-Taste gedrückt, so kann der Text nur mit dem Menüpunkt "Save-file-as" abgespeichert werden.

Read-file Ctrl-X, Ctrl-R

Mit diesem Menüpunkt wird der Inhalt einer Datei in den aktuellen Textspeicher geladen. Ist in diesem Puffer ein Text, der verändert wurde, so erscheint nach der Aufforderung zur Eingabe des Namens der zu ladenden Datei "Read file:" die Frage "Discard Changes" (Änderungen verwerfen). Wird diese mit "Y" beantwortet, so wird der Text gelöscht, ohne ihn vorher abzuspeichern. Wird die Frage mit "N" beantwortet, so unterbleibt das Laden der neuen Datei. Als Dateiname ist der vollständige Pfad anzugeben, wenn sich die Datei nicht im aktuellen Verzeichnis befindet. Wollen Sie keine Datei einlesen, so können Sie den Befehlsmodus durch drücken der Return-Taste wieder verlassen.

Visit-file Ctrl-X, Ctrl-V

Es gibt bei MEMcas eine ganz besondere Art, Daten einzulesen - durch den Besuch einer Datei mit dem Menüpunkt "Visit-file". Hierbei kann ein Text geladen und dargestellt, verändert und neu abgespeichert werden, wie bei normalen Texten auch. Ein Unterschied besteht jedoch darin, daß der Text einen eigenen Textspeicher bekommt, der aktuelle Buffer wird nicht überschrieben. Der so geladenen Text kann ebenfalls geändert und abgespeichert werden, eine Sicherheitsabfrage unterbleibt jedoch, wenn der Text verändert wurde und vor dem Löschen des Puffers nicht abgespeichert wurde.

Insert-file Ctrl-X, Ctrl-I

Nach Auswahl dieser Funktion werden Sie in der untersten Zeile aufgefordert, den Namen einer Datei einzugeben, dessen Inhalt an der aktuellen Cursorposition eingefügt wird. Befindet sich die Quelldatei nicht im aktuellen Verzeichnis, so ist der komplette Pfad anzugeben.

Save-file Ctrl-X, Ctrl-S

Wurde der Text des aktuellen Buffers von einer Datei eingeladen, so können inzwischen vorgenommene Änderungen in diese Datei wieder abgespeichert werden. Der Inhalt der ursprünglichen Datei geht dadurch jedoch verloren. Wurde keine Datei geladen, so erscheint in der untersten Zeile die Meldung "No file name", der Text wird nicht gespeichert.

Wurde mit dem Menüpunkt "Rename" dem Puffer ein anderer Name gegeben oder der Puffername gelöscht, so kann diese Funktion nicht ausgeführt werden, da MEmacs nicht weiß, unter welchen Namen der Text abgespeichert werden soll. In diesem Fall wird der Anwender jedoch nicht davon informiert, daß der Text nicht gespeichert wurde.

Save-as-file Ctrl-X, Ctrl-W

Hiermit kann der Text des aktuellen Speichers auf jeden Fall gespeichert werden. MEmacs fordert Sie in der letzten Zeile dazu auf, den Namen einzugeben, unter der der Text gespeichert werden soll: "Write file:". Es kann auch ein Pfadname eingegeben werden, wenn die Daten nicht im aktuellen Verzeichnis gespeichert werden sollen. Achtung: Wenn eine Datei mit dem angegebenen Namen bereits existiert, so wird deren Inhalt gnadenlos überschrieben.

Wenn Sie bei der Aufforderung zur Eingabe eines Dateinamens lediglich die Return-Taste drücken, so unterbleibt jegliches Speichern des Textes, der Stern zur Kennzeichnung, ob der Text seit dem letzten Speichern verändert wurde, bleibt ebenfalls stehen.

Save-mod Ctrl-X, Ctrl-M

Mit diesem Menüpunkt können alle Textspeicher, die geändert (also MODifiziert) wurden in die jeweiligen Quelldateien zurückgeschrieben werden. Vorsicht ist jedoch bei Textpuffer angesagt, deren Namen mit "Rename" geändert wurden.

Save-exit Ctrl-X, Ctrl-F

Mit diesem Menüpunkt werden alle modifizierten Textspeicher in die entsprechenden Quelldateien geschrieben und anschließend das Programm MEmcas verlassen. Die gleiche Wirkung hätte die Ausführung von "Save-file" und "Quit" hintereinander. Hier muß ebenfalls besonders darauf geachtet werden, daß Dateien, deren Puffername durch "Rename" geändert wurde, nicht korrekt gespeichert werden können.

New-Cli**Ctrl- -**

Dieser Menüpunkt öffnet ein Shell-Fenster namens "Spawn Window" im Bildschirm des MicroEmacs. Sie können darin arbeiten, wie in einer normalen Shell, der Editor ist jedoch solange auf Eis gelegt, solange das Fenster geöffnet ist. Mit dem Befehl "ENDCLI" kann das Fenster wieder geschlossen und die Arbeit im Editor fortgeführt werden.

Cli-Command Ctrl-X, !

Innerhalb von MEmacs können auch AmigaDOS-Befehle aufgerufen werden, indem der Menüpunkt Cli-Command gewählt wird. MEmacs springt dadurch in den Befehlsmodus und fordert den Anwender in der letzten Zeile des Bildschirms dazu auf, einen AmigaDOS-Befehl einzugeben. Der Befehl ist genau so einzugeben, wie er in einem Shell-Fenster eingegeben worden wäre. Wenn der Befehl bei seiner Ausführung auch Ausgaben produziert, so werden diese in einem MEmacs-Fenster angezeigt und gleichzeitig im Textpuffer "Spawn.output" gespeichert.

Quit**Ctrl-C**

Mit diesem Menüpunkt wird MEmacs verlassen. Wurden seit dem letzten Speichern ein oder mehrere Puffer verändert, so werden Sie davon informiert und gefragt, ob Sie MEmacs beenden wollen, obwohl diese Text nicht gespeichert wurden: "Modified buffers exist, do you really want to exit? [y/n]" Wenn Sie diese Frage mit "Y" beantworten, so werden die Änderungen verworfen und der Editor verlassen. Wird jedoch "n" eingegeben oder nur die Return Taste drücken, so wird der Editor nicht verlassen.

About

Dieser Menüpunkt ist auch von anderen Programmen bekannt. Es werden lediglich Informationen zum Programmautor und den Programmrechten ausgegeben. Weitere Funktionen hat dieser Punkt nicht.

3.7.2.2 Das Menü "Edit"

Die Punkte des Menüs "Edit" beziehen sich in erster Linie auf die Textspeicher.

Kill-region Ctrl-W

Mit diesem Befehl wird der Bereich des Textes zwischen der Markierung (siehe obige Anmerkung zu "Mark" und zu "Set-mark" weiter unten) und der aktuellen Cursorposition aus dem Text entfernt und in den sogenannten "Kill-Buffer" zwischengespeichert. Mit der Funktion "Yank" kann dieser Text wieder an der Stelle des Cursors eingefügt werden.

Wird die Funktion "Kill-region" wiederholt ausgeführt, so wird der "Kill-Buffer" durch die neu ausgeschnittenen Texte überschrieben und der alte nicht durch die neuen Textblöcke erweitert.

Yank

Ctrl-Y

PAG Sandini

Mit diesem Menüpunkt wird der Inhalt des "Kill-Buffer" an der aktuellen Cursorposition eingefügt, der Inhalt des Kill-Buffer wird jedoch weiterhin beibehalten, so daß der gleiche Text an mehreren Stellen und eventuell auch in mehreren Textspeichern eingefügt werden kann. Es existiert nur ein Kill-Buffer, wenn auch mehrere Textpuffer vorhanden sind.

Set-mark

Ctrl-Alt-@ oder Esc -

Hiermit wird die aktuelle Cursorposition als Anfang eines Blocks gespeichert ("Mark"), eine eventuell vorhergehende andere Definition wird dadurch aufgehoben. Das Zeichen unter dem Cursor befindet sich dann jeweils schon innerhalb eines Blocks. Anstelle der etwas umständlicheren Tastaturkombination Ctrl-Alt-@ kann auch die Kombination "Esc -" verwendet werden. Wurde eine Marke gesetzt, so erscheint in der letzten Zeile des Fensters die Meldung "[Mark set]". Der Bereich zwischen der mit "Set-mark" vereinbarten Marke und der aktuellen Cursorposition wird auch als Block bezeichnet, die aktuelle Cursorposition selbst als Dot. Jeder Textspeicher hat seine eigenen Marks und Dots.

Copy-region Esc, W

Der Bereich zwischen der mit Set-mark definierten Marke und der aktuellen Cursorposition wird in den Kill-Buffer übernommen, jedoch nicht aus dem Text entfernt. Der bisherige Inhalt des Kill-Buffers wird dadurch überschrieben.

Upper-region Ctrl-X, Ctrl-U

Mit dieser Funktion werden alle Buchstaben des gewählten Blocks (also zwischen Mark und Dot) in Großbuchstaben umgewandelt. Wie eingangs schon erwähnt, ist dies bei den sprachspezifischen Umlauten (wie beispielsweise ä, ö oder ü) nicht möglich, da diese von MicroEmacs als Sonderzeichen (auf einer Stufe mit den Satzzeichen) behandelt werden.

Lower-region Ctrl-X, Ctrl-L

Analog zur Funktion "Upper-region" werden hier alle Buchstaben eines Blocks klein gemacht. Auch hier können Umlaute nicht umgewandelt werden.

List-buffers Ctrl-X, Ctrl-B

Durch Auswahl dieses Menüpunkts wird das MEmacs-Fenster geteilt und die Liste der verwalteten Textspeicher angezeigt. In dieser tabellenartigen Liste werden folgende Informationen wiedergegeben: In der ersten Spalte unter dem Buchstaben "C" steht ein Stern in der Zeile eines Pufferspeichers, wenn dessen Inhalt verändert wurde, aber noch nicht abgespeichert ist. Es folgt eine Zahl, die die Größe des Textes in Zeichen angibt, daran schließt sich der Name des Textspeichers an, gefolgt vom Namen der Quelldatei, falls diese bekannt ist.

Obwohl die Liste der verwalteten Textpuffer selbst in einem Fenster namens "[List]" steht, handelt es sich hier um keinen vergleichbaren Textpuffer, er kann nicht ausgewählt werden.

Select-buffer Ctrl-X, B

Sind MEmacs mehrere Textspeicher bekannt, so kann der Inhalt eines ande-

ren Textspeichers, dessen Name in der letzten Zeile durch den Anwender eingegeben ist, im aktuellen Fenster angezeigt werden. Ist kein Puffer mit dem angegebenen Namen vorhanden, so wird ein neuer Puffer mit diesem Namen geschaffen. Wird statt der Eingabe eines Puffernamens lediglich die Return-Taste gedrückt, wirkt sich dieser Menüpunkt nicht weiter aus.

Insert-buffer Esc, Ctrl-Y

Mit diesem Menüpunkt kann der Inhalt eines anderen Textspeichers an der aktuellen Cursorposition im aktuellen Text eingefügt werden. Dazu müssen Sie in der letzten Zeile nach der Aufforderung "Insert buffer:" den Puffernamen eingeben.

Kill-buffer Ctrl-X, K

Ab und an ist es notwendig, einen bestehenden Textspeicher wegen mangelndem Speicher zu löschen, was mit dieser Funktion geschieht. Dazu muß wieder der Name des Puffers eingegeben werden, der entfernt und dessen Speicher damit wieder verfügbar gemacht werden soll. Der gerade angezeigte Textspeicher kann nicht entfernt werden.

Justify-buffer Ctrl-X, J

Hier werden alle Leerzeichen und Tabulatoren im aktuellen Textspeicher am linken Rand gelöscht, der Text wird linksbündig gesetzt.

Redisplay Ctrl-L

Gerade nach der Bearbeitung eines Textes mit der Funktion "Justify-buffer" kann es vorkommen, daß die Änderungen zwar vorgenommen wurden, aber nicht angezeigt werden. Um dies jedoch zu ermöglichen, kann MicroEmacs mit dem Menüpunkt "Redisplay" gezwungen werden, alle Fenster neu aufzubauen. Dabei werden eventuelle Fehler beseitigt.

Quote-char Ctrl-Q

Oft werden in einem Text Zeichen benötigt, die man nirgendwo auf der Tastatur finden kann, weil sie entweder so versteckt sind, die eigentliche

Tastenkombination anders belegt wurde, oder gar nicht durch die Tastatur erzeugt werden kann. Gerade bei den Ctrl-Kombinationen ist es oft nicht möglich, das eigentliche Zeichen einer bei gleichzeitigem Drücken von Ctrl und einer weiteren Taste zu erreichen, da MEMacs dies fälschlicherweise als Funktionsaufruf wertet. Dieser Menüpunkt schafft Abhilfe.

Wenn man unmittelbar vor der Eingabe des Sonderzeichens den Menüpunkt "Quote-char" direkt oder über die Tastenkombination Ctrl-Q aufruft, so wird das nächste Zeichen direkt übernommen, ohne es als irgend einen Befehl zu werten. Wichtig ist dabei, daß immer nur ein Zeichen unverändert übernommen wird, schon das nächste wird wieder auf Sonderaufgaben hin überprüft.

Indent Ctrl-J

Dieser Menüpunkt wirkt in etwa wie das Drücken der Return-Taste, der Cursor springt in die nächste Zeile, der Teil der sich in der vorherigen Zeile rechts vom Cursor befand, wird abgeschnitten und mitgenommen. Der wesentliche Unterschied besteht darin, daß die neue Zeile automatisch soweit eingerückt wird, wie die unmittelbar darüberliegende Zeile eingerückt war. Dabei werden Leerzeichen und Tabulatoren in ihrer Anzahl und Position genauso übernommen, wie sie in der darüberliegenden Zeile vorkommen.

Transpose Ctrl-T

Diese Funktion vertauscht das Zeichen direkt unter dem Cursor mit dem links davon stehenden Zeichen.

Cancel Ctrl-G

Mit diesem Menüpunkt wird eine laufende Funktion, die über das Menü gewählt wurde, augenblicklich unterbrochen.

3.7.2.3 Das Menü "Window"

So überraschend gravierend wie sich die Fenster des MicroEmacs von denen des Betriebssystems unterscheiden, so ungewohnt können die Funktionen arbeiten, die auf diese wirken.

One-window Ctrl-X, 1

Werden gleichzeitig mehrere Fenster gezeigt (der Bildschirm ist horizontal unterteilt), so kann mit dieser Funktion das aktuelle Fenster auf die Größe des gesamten Bildschirms gebracht werden. Die anderen vorhandenen Fenster werden dabei entfernt, deren Inhalt bleibt jedoch in den Textspeichern weiterhin verfügbar.

Split-window Ctrl-X, 2

Dieser Menüpunkt bewirkt genau das Gegenteil, hier wird bei nur einem vorhandenen Fenster der Bildschirm horizontal unterteilt, es entstehen zwei Fenster. Sind es bereits schon mehrere Fenster, so wird das aktuelle weiter unterteilt. Die maximale Anzahl der darstellbaren Fenster ist lediglich durch die Anzahl der darstellbaren Zeilen in einem Bildschirm begrenzt.

Sogar bei Gewaltanwendung mit einem MEmacs-Fenster auf einer Workbench mit mehr als 2400 Pixel Höhe und einem überdimensionierten Fenster konnte ich keine Grenze des Machbaren erreichen - es ist nur eine Frage des verfügbaren Speichers. Was ich jedoch festgestellt habe, ist die Eigenschaft des Editor - sofern er in einem Workbench-Fenster läuft - daß er nach dem Ändern der Fenstergröße nur noch das MEmacs-"Fenster" des aktuellen Textspeichers anzeigt, alle weiteren vorher dargestellten "Fenster" werden eliminiert.

Next-window Ctrl-X, N

Wenn im Bildschirm mehrere Fenster zu sehen sind, so springt der Cursor in das nächstniedrigere, dessen Inhalt dadurch zum aktuellen Textspeicher erklärt wird. Befindet sich der Cursor bereits im untersten Fenster, so wird er in das erste verfrachtet.

Prev-window Ctrl-X, P

Dieser Menüpunkt bewirkt wiederum genau das Gegenteil seines Vorgänger, hier springt der Cursor ein Fenster nach oben, womit dessen Textspeicher als neuer aktueller Speicher gilt. Ist der Cursor bereit im obersten Fenster, so springt er in das unterste.

Expand-window Ctrl-X, Z

Mit diesem Menüpunkt wird das aktuelle Fenster um eine Zeile vergrößert, sofern dies überhaupt möglich ist. Ist dies technisch nicht realisierbar, erscheint in der letzten Zeile eine entsprechende Meldung. Durch diese Aktion wird gleichzeitig das untere Fenster um eine Zeile kleiner.

Shrink-window Ctrl-X, Ctrl-Z

Diese Funktion verkleinert das aktuelle Fenster um eine Zeile, das Fenster unterhalb wird um eine Zeile reicher. Auch diese Änderung ist nicht immer durchführbar, es gibt auch hier gegebenenfalls eine Meldung in der letzten Zeile darüber Auskunft.

Next-w-page Esc, Ctrl-V

Mit diesem Menüpunkt wird der Text des Fensters um eine Seite weiter geblättert, das gerade nicht aktiv ist (in dem kein Cursor ist). Ist der Bildschirm in mehr als nur zwei Fenster unterteilt, so wird der Inhalt des jeweils nächsten (nach unten hin) Fensters um eine Seite vorgeblättert, ist dagegen der Cursor im untersten Fenster zu sehen, so wird der Inhalt des obersten verschoben.

Prev-w-page Ctrl-X, V

Hiermit wird im folgenden Fenster (nicht im aktuellen) der Text um eine Seite zurückgeblättert. Ist der Cursor gerade im untersten Fenster, so wird der Inhalt des obersten Fenster zurückgeblättert.

3.7.2.4 Das Menü "Move"

Das Menü "Move" stellt Routitionen zur Verfügung, mit denen der Cursor im aktuellen Textspeicher schnell bewegt werden kann.

Top-of-buffer Esc, <

Der Cursor springt an den Anfang des aktuellen Textspeichers.

End-of-buffer Esc, >

Der Cursor springt an das Ende des aktuellen Textspeichers.

Top-of-window Esc,,

Der Cursor springt an den Anfang des aktuellen MEMacs-Fenster.

End-of-window Esc,.

PAG Sandini

Der Cursor springt an das Ende des aktuellen MEMacs-Fenster.

Goto-line Ctrl-X, Ctrl-G

Durch diesen Menüpunkt erscheint in der letzten Zeile die Aufforderung, die Zeile einzugeben, in die der Cursor springen soll. Wenn Sie eine Zahl eingeben, so wird der Cursor an den Anfang der entsprechenden Zeile gesetzt. Ist die eingegebene Zahl jedoch größer als die Gesamtzahl der Zeilen des Textes, so wird der Cursor an den Beginn der letzten Zeile des Textes gesetzt.

Swap-dot&mark Ctrl-X, Ctrl-X

Die aktuelle Position des Cursor wird als Marke gespeichert, der Cursor springt selbst an die Stelle, die zuvor als Marke gesetzt wurde. Wurde bislang im aktuellen Textspeicher noch keine Marke gesetzt, so wird die Funktion nicht ausgeführt, es erscheint die Meldung "No mark in this window". Diese Funktion ermöglicht ein sehr schnelles hin- und herspringen zwischen zwei bestimmten Stellen im Text.

Next-page **Ctrl-V**

Der Text wird um so viele Zeile vorgerückt, wie dem aktuellen Fenster zur Verfügung stehen. Der Cursor selbst bleibt an seiner Stelle stehen, es sei denn, er kommt in einer leeren Zeile zu stehen. In diesem Fall springt der Cursor an den Anfang der entsprechenden Zeile.

Prev-page **Esc, V**

Der Text wird um so viele Zeilen zurückgeschoben, wie dem aktuellen Fenster zur Verfügung stehen. Der Cursor selbst bleibt an seiner Stelle stehen, es sei denn, er kommt in einer leeren Zeile zu stehen. In diesem Fall springt der Cursor an den Anfang der entsprechenden Zeile.

Next-word **Esc, F**

Der Cursor springt auf das erste Zeichen hinter dem aktuellen oder nächsten Wort, das kein Buchstabe oder keine Ziffer ist.

Previous-word **Esc, B**

Der Cursor springt auf das erste Zeichen vor dem aktuellen oder nächsten Wort links neben dem Cursor, das kein Buchstabe oder keine Ziffer ist.

Scroll-up **Ctrl-Z**

Diese Funktion verschiebt den Text um eine Zeile nach oben.

Scroll-down **Esc, Z**

Hiermit wird der Text um eine Zeile nach unten verschoben.

3.7.2.5 Das Menü "Line"

Dieses Menü enthält Funktionen, die zeilenweise arbeiten.

Open-line Ctrl-O

Dieser Menüpunkt bewirkt das gleiche, als wenn die Return-Taste gedrückt worden wäre - die Zeile wird an der Position des Cursors geteilt, die Zeichen unter dem Cursor und rechts davon werden in eine neue Zeile gesetzt. Der Unterschied zur Return-Taste besteht jedoch darin, daß der Cursor nicht in die nächste Zeile springt, sondern an der Stelle bleibt, an der die Funktion aufgerufen wurde.

Kill-line Ctrl-X, Ctrl-D

Die Zeile, in der sich der Cursor befindet, wird aus dem Text entfernt. Die Zeile wird in den Kill-Buffer übernommen und kann damit Yank wieder eingebaut werden.

Kill-to-eol Ctrl-K

Dieser Menüpunkt löscht alle Zeichen rechts vom Cursor einschließlich des Zeichens unter dem Cursor und übernimmt den Text in den Kill-Buffer, der mit Yank an beliebiger Stelle wieder eingefügt werden kann.

Start-of-line Ctrl-A

Der Cursor wird an den Anfang der aktuellen Zeile gesetzt.

End-of-line Ctrl-E

Der Cursor springt hinter das Zeichen am rechten Rand der Zeile, auch wenn das Ende nicht dargestellt wird, sollte die Zeile zu lang sein.

Next-line Ctrl-N

Der Cursor springt eine Zeile tiefer.

Previous-line Ctrl-P

Der Cursor springt eine Zeile höher.

Line-to-top Esc, !

Der Text wird nach oben verschoben, so daß die Zeile, in der sich der Cursor befindet, die erste Zeile im aktuellen Fenster ist.

Delete-blanks Ctrl-X, Ctrl-O

Es werden alle Leerzeilen ab der Cursorposition gelöscht, bis eine Zeile auftaucht, die wieder Zeichen enthält.

Show-Line# Ctrl-X, =

Hiermit kann man sich Informationen über die aktuelle Position des Cursor holen. Angegeben wird die Zeilen- und Spaltennummer sowie der Prozentsatz des Textes, der oberhalb des Cursors liegt. Befindet sich der Cursor in der letzten Zeile, so ist 100% des Textes oberhalb, ist er beispielsweise am Anfang des Textes, so ist die Position bei 0%.

3.7.2.6 Das Menü "Word"

Delete-forw Esc, D

Es werden alle Zeichen rechts vom Cursor - beginnend mit dem, das unter dem Cursor liegt - gelöscht, bis ein Leerzeichen oder Satzzeichen erscheint. Das Leerzeichen oder Satzzeichen selbst wird nicht mehr gelöscht.

Delete-back Esc,H oder Esc, Del

Hiermit werden alle Zeichen links vom Cursor gelöscht, bis der Anfang des Wortes erreicht wird.

Upper-word Esc, U

Die Buchstaben eines Wortes werden ab der aktuellen Cursorposition bis zum Wortende in Großbuchstaben verwandelt.

Lower-word Esc, L

Die Buchstaben eines Wortes werden ab der aktuellen Cursorposition bis zum Wortende in Kleinbuchstaben verwandelt.

Cap-word Esc, C

Der Buchstabe unter dem Cursor wird in einen Großbuchstaben umgewandelt, alle rechts vom Cursor liegenden Buchstaben werden bis zum Wortende in Kleinbuchstaben verwandelt.

Switch-case Esc, ^

Ab der aktuellen Cursorposition werden bis zum Wortende alle Kleinbuchstaben in Großbuchstaben umgewandelt und umgekehrt.

3.7.2.7 Das Menü "Search"

Dieses Menü stellt verschiedene Funktionen zur Verfügung, mit denen der Text nach dem Auftreten bestimmter Zeichenketten durchsucht werden kann, oder auch durch andere ersetzt werden können. Bei der Suche nach der eingegebenen Zeichenkette werden Groß- und Kleinbuchstaben nicht unterschieden, wird die Zeichenkette jedoch ersetzt, so wird die Schreibweise der Eingabe übernommen.

Search-forward Ctrl-S oder Ctrl-X, S

Nach Auswahl dieses Menüpunkts fordert MEmacs den Anwender in der letzten Zeile auf, die Zeichenkette einzugeben, nach der gesucht werden soll. Sobald die Eingabe durch die Return-Taste abgeschlossen wird, sucht

MEmacs die Zeichenkette ab der aktuellen Cursorposition bis zum Ende des Textes. Sobald die erste Zeichenkette gefunden wird, die mit der eingegebenen übereinstimmt, springt der Cursor hinter die entsprechende Zeichenkette und bleibt dort stehen. Groß- und Kleinbuchstaben werden nicht berücksichtigt. Wird keine passende Zeichenkette gefunden, so erscheint die Meldung "Not found".

Search-backward Ctrl-R oder Ctrl-X, R

Analog dem Menüpunkt "Search-forward" werden Sie hier aufgefordert, die Zeichenkette einzugeben, die gesucht werden soll. Im Gegensatz zum vorherigen Menüpunkt wird der Text hier ab der aktuellen Cursorposition in Richtung Anfang des Textes durchsucht.

Search-replace Esc, R

Dieser Befehl arbeitet am Anfang wie der Menüpunkt "Search-forward". Wird die Zeichenkette zum ersten Mal gefunden, erscheint in der letzten Zeile die Aufforderung, die Zeichenkette einzugeben, durch die die alte ersetzt werden soll. Wird die Eingabe mit der Return-Taste abgeschlossen, so werden sämtliche Zeichenketten, die der zu suchenden entsprechen, durch die neue ersetzt. Nach erfolgter Ausführung erscheint in der letzten Zeile die Meldung "Replaced ## occurrences" (habe die Zeichenkette ##-mal ersetzt).

Query-s-r Esc, Q

Dieser Menüpunkt gleicht dem vorhergehenden "Search-replace" im Wesentlichen. Die Unterschiede sind schnell erläutert. Bei "Query-s-r" wird der Anwender gleich nach Eingabe der Suchzeichenkette aufgefordert, die Ersatzzeichenkette einzugeben. Bei dem ersten Auftreten der Suchzeichenkette erscheint in der letzten Zeile die Frage "Change string? [y/n/c/^G]?". Sie können hier entscheiden, ob nur diese Zeichenkette ersetzt werden soll (Y eingeben) oder nicht (N eingeben), oder diese und alle anderen (C eingeben). Durch die Eingabe von Ctrl-G wird die Funktion abgebrochen. Auch hier wird nach der Ausführung des Befehls angezeigt, wieviele Zeichenketten ersetzt wurden.

Fence-match Esc, Ctrl-F

Der Text wird nach dem nächsten Auftreten des gleichen Zeichens durchsucht, daß sich unter dem Cursor befindet. Handelt es sich jedoch um eine Klammer - egal ob rund, eckig, geschweift oder spitz - so wird das entsprechende Gegenstück gesucht. Ist es eine öffnende, so wird die zugehörige schließende Klammer gesucht, befindet sich eine schließende unter dem Cursor, so wird die zugehörige öffnende Klammer gesucht.

3.7.2.8 Das Menü "Extras"

Die Routinen, die das Menü "Extras" bereitstellt, erleichtern die Arbeit mit MEMacs, da komplexe Befehle und oft wiederkehrende Befehlsabläufe automatisiert werden können. Bei vielen dieser Menüpunkte muß vor der Auswahl durch den Menüpunkt "Set-arg" eine Zahl eingegeben werden, bevor der eigentliche Punkt ausgewählt werden kann.

Set-arg

Ctrl-U

PAG Sandini

Nach Auswahl dieses Menüpunktes werden Sie in der letzten Zeile mit der Meldung "Arg: " aufgefordert, eine Zahl einzugeben. Wird dieser Befehl erneut ausgeführt, so wird die eingegebene Zahl mit 4 multipliziert. MEMacs akzeptiert jedoch nur die Eingabe von Ziffern über die normale Tastatur, wird eine Taste des Zehner-Blocks gedrückt, so wird der Menüpunkt "Set-arg" verlassen, wie auch durch jede andere Taste, die keine Ziffer darstellt.

Viele der nachfolgenden Menüpunkte greifen auf die hier angegebene Zahl zurück. Wird jedoch anstatt eines weiteren Menüpunktes eine andere Taste gedrückt, so wird die Aktion, die durch die gedrückte Taste ausgelöst wird, sooft wiederholt, wie als Arg eingegeben wurde. Geben Sie beispielsweise "Arg: 500" ein und drücken nach der zweiten Null die Taste "G", so werden an der Position des Cursors 500 G's eingefügt.

Set

Esc, S

Dieser Menüpunkt erlaubt es dem Anwender, diverse Parameter des MEMacs zu verändern. In der letzten Zeile des Bildschirms erscheint die Eingabeauf-

forderung "Set what: ". Hier können Sie eines der folgenden Schlüsselwörter eingeben, bei den meisten erscheint eine weitere Aufforderung, den entsprechenden Wert einzugeben, einige wenige sind jedoch selbst Schalter und benötigen keine weiteren Angaben. Bei den Schlüsselwörtern "Left", "Right", "Tab" und "Indent" können durch ein Leerzeichen getrennt breits die entsprechenden Zahlen bei der Aufforderung "Set what:" eingegeben werden. Geschieht dies nicht, so erscheint eine gesonderte Eingabeaufforderung.

Screen Wurde MEmacs mit dem Argument "Opt W" gestartet, so daß der Editor ein Fenster auf der Workbench besitzt, so wird dieses geschlossen und MEmacs bekommt einen eigenen Bildschirm. Wird dagegen der Text bereits auf einem eigenen Bildschirm dargestellt, so wird dieser entfernt und durch ein Fenster in der Workbench ersetzt. Dieses Schlüsselwort benötigt keine weiteren Eingaben.

Interlace Mit diesem Schlüsselwort kann das sogenannte Zeilensprungverfahren des Bildschirmes aus- bzw. eingeschaltet werden. Ist es eingeschaltet, können zwar möglicherweise mehr Zeilen dargestellt werden, der Anwender muß dafür aber ein sehr starkes Flimmern des Bildes in Kauf nehmen. Bei manchen höheren Bildschirmauflösungen kann dieser Schalter ohne Auswirkungen bleiben. Weitere Werte werden nicht gefordert.

Mode Hier müssen weitere Angaben nach die Aufforderung "Mode: "gemacht werden. Es sind hier die Schalter "Cmode" (hilft bei der Erstellung von C-Programmen) oder "Wrap" (schaltet den automatischen Zeilenumbruch ein, dh. ragt ein Wort über das Zeilenende hinaus, so wird es komplett in die neue Zeile gesetzt) möglich. Es können beide Modi gleichzeitig verwendet werden, jedoch nicht gleichzeitig eingegeben werden. Um einen neuen Modus hinzuzufügen, muß dem Schlüsselwort das Zeichen "+" beim Löschen das Zeichen "-" vorangestellt werden.

Left Hier ist eine weitere Zahl einzugeben, die den linken Rand festlegt.

Right	Hier ist eine weitere Zahl einzugeben, die den rechten Rand festlegt.
Tab	Hiermit kann angegeben werden, wieviele Leerzeichen durch das Drücken der Tabulatortaste eingefügt werden sollen.
Indent	Durch die hier anzugebende Zahl wird festgelegt, wieweit ein Abschnitt im Modus "Cmode" eingerückt werden soll. "Cmode" rückt bei verschiedenen Verschachtelungsebenen in C-Programmen automatisch ein.
Case	Mit diesem Schalter, der keine weiteren Werte benötigt, kann dem Editor MEMacs mitgeteilt werden, daß er bei der Suche nach Zeichenketten die Groß- und Kleinschreibung berücksichtigen soll. Dieser Schalter ist ein Wechselschalter, dh. er schaltet den Modus ein, wenn er vorher ausgeschaltet war und umgekehrt. Beim Programmstart wird die Groß- und Kleinschreibung ignoriert.
Backup	Diese Vorgabe benötigt eine der drei folgenden Schalter als weitere Werte: ON wird der Text aus einer Datei gelesen, so wird deren Namen die Erweiterung ".bak" angehängt, so daß die ursprüngliche Version erhalten bleibt, wenn Änderungen abgespeichert werden. Die Sicherheitskopie wird im Verzeichnis T: gespeichert. SAFE Mit diesem Schalter wird MEMacs angewiesen, vor dem Erstellen der Sicherheitskopie zu prüfen, ob nicht schon eine gleichnamige Datei vorhanden ist. Ist dies der Fall, so wird keine Sicherheitskopie erstellt. OFF Es wird keine Sicherheitskopie erstellt. Dieser Modus ist voreingestellt.

Start-macro Ctrl-X, (

Dieser Menüpunkt sagt dem Editor MEMacs, daß er sich alle folgenden Eingaben (auch Auswahl von Menüpunkten) in der entsprechenden Reihenfolge merken soll. Alle Befehle, die zwischen dem Menüpunkt "Start-macro" und "Stop-macro" ausgeführt werden, werden als ein sogenanntes Macro gespeichert. Die Befehle werden jedoch trotzdem ausgeführt. Durch den Punkt "Stop-macro" wird dieser "Merk-Modus" beendet.

Stop-macro Ctrl-X,)

Alle nachfolgenden Befehle werden nicht mehr in das Makro aufgenommen.

Execute-macro Ctrl-X, E

Alle Befehle, die zwischen "Start-macro" und "Stop-macro" ausgeführt wurden, werden automatisch in der entsprechenden Reihenfolge erneut ausgeführt.

PAG Sandini

Set-key Ctrl-X, Ctrl-K

Mit diesem Menüpunkt können Sie die Funktionstasten mit und ohne Shift-Taste, die Help-Taste und die des Zehner-Blocks nach eigenen Bedürfnissen mit Macro-Befehlen belegen. Jede Taste kann bis zu achzig Zeichen fassen.

Nach der Auswahl dieses Menüpunktes werden Sie in der letzten Zeile mit der Meldung "key to define:" aufgefordert, die Taste zu drücken, die mit dem gewünschten Makro belegt werden soll. Es erscheint die Meldung "def: [bisherige Befehle]: " Hier werden die neuen Befehle eingegeben, mit der die Taste belegt werden soll, die Eingabe ist mit der Return-Taste abzuschließen. Bei der Eingabe von Befehlen mit Sonderzeichen wie etwa "Esc" muß mit dem Befehl Quote-char oder Ctrl-Q MEMacs mitgeteilt werden, daß das jeweils nächste Zeichen unverändert in den Befehl übernommen werden soll.

Die Voreinstellungen entnehmen Sie der folgenden Tabelle:

Taste	Funktion	Tastenfolge
F1	Zeile kopieren	Ctrl-A, Ctrl-K, Ctrl-Y, Ctrl-M, Ctrl-Y
F2	Zeile löschen	Ctrl-X, Ctrl-D
F3	Tastaturmakro ausführen	Ctrl-X, E
F4	eine Seite vorblättern	Ctrl-V
F5	eine Seite zurückblättern	Esc-V
F6	Fenster teilen	Ctrl-X, 2
F7	ein Fenster	Ctrl-X, 1
F8	eine Zeile nach oben rollen	Ctrl-Z
F9	eine Zeile nach unten rollen	Esc, Z
F10	Text speichern und beenden	Ctrl-X, Ctrl-F
Help	Zeile einfügen	Ctrl-J
Enter	Zeile einfügen	Ctrl-J

Alle Tasten den Zehner-Blocks sind voreinstellungsgemäß mit den Zeichen belegt, die sie üblicherweise enthalten nämlich ihre eigenen Ziffern. Doch können diese auch mit Befehlen belegt werden.

Reset-keys Esc, K

Dieser Menüpunkt macht alle benutzerseitigen Änderungen der Tastenbelegungen rückgängig. Die Tasten erhalten die voreingestellten Funktionen.

Execute-file Esc, E

Dieser Menüpunkt ermöglicht es dem Anwender, MEmacs-Befehle in einer Datei zu sammeln und an dieser Stelle auszuführen. Dazu wird er in der untersten Zeile aufgefordert, den Namen der Datei mit den MEmacs-Befehlen einzugeben.

Die MEmacs-Befehle sind als Tastaturkürzel wie sie in den Menüs stehen, einzugeben, gefolgt von eventuell benötigten weiteren Daten.

Bei der Belegung der Funktionstasten, sind diese durch "F#" anzugeben, sollen die Funktionstasten in Verbindung mit der Shift-Taste belegt werden, so muß die Nummer der Funktionstaste mit der Zahl 10 addiert werden. Die Angabe "F14" bezeichnet beispielsweise die Tastenkombination Shift-F4.

Execute-line Ctrl-[, Ctrl-[oder Esc, Esc

Der Editor MEmacs aktiviert den Befehlsmodus, es erscheint die Eingabeaufforderung: "execute line:". Hier muß ein beliebiger Menüpunkt in Textform gefolgt von eventuell weiteren benötigten Werten eingegeben werden.

3.7.3 Weitere MEmacs-Befehle

MEmacs kennt noch einige weitere Befehle, die jedoch nicht im Menü zu finden sind, sondern nur über die Tastatur oder eine entsprechende MEmacs-Befehls-Datei benutzt werden können.

Describe-key Esc, Ctrl-D

Nach Eingabe dieses Befehles wird der Anwender aufgefordert, die Tastenkombination eines anderen Menüpunktes anzugeben. Die Eingabe muß durch die Return-Taste abgeschlossen werden. Handelt es sich bei der angegebenen Tastenkombination um eine gültige MEmacs-Funktion, so wird deren Bezeichnung und die entsprechende Tastenkombination angezeigt.

Bind-key Esc, Ctrl-B

Mit diesem Menüpunkt können den einzelnen Funktionen des Menüs neue Tastenkombinationen zugewiesen werden. Dazu muß zuerst einmal der Name des entsprechenden Menüpunktes eingegeben werden und anschließend die neue Tastaturkombination. Durch den obigen Befehl "Describe-key" kann anschließend geprüft werden, ob die Aktion erfolgreich war oder nicht.

Unbind-key Esc, Ctrl-U

Mit diesem Befehl sollten definierte Tastaturkombinationen aufgelöst werden. Leider konnte ich bei meinen Tests jedoch diesen Befehl nicht aufrufen, da die Kombination Ctrl-U die Eingabe der Taste Esc wieder löschte.

Echo Esc, Ctrl-E

Nach diesem Befehl werden die folgenden Befehl nicht ausgeführt, sondern deren Zeichen in der untersten Zeile dargestellt. Dies ist eine große Hilfe vor allem beim Erstellen von Befehlsdateien für MEMacs, da hier direkt abgelesen werden kann, wie der eine oder andere Befehl eingegeben werden muß.

Move to Edge of Window Shift-Cursor

Wird eine der Cursor-Tasten in Verbindung mit der Shift-Taste gedrückt, so springt der Cursor in Abhängigkeit der gedrückten Cursor-Taste an den linken, rechten, oberen oder unteren Rand des Fensters, falls der Text entsprechend groß ist, andernfalls an das Ende oder den Beginn des Textes bzw. der Zeile.

Delete the next Character Ctrl-D
Dieser Befehl löscht das Zeichen unter dem Cursor.

Delete the previous Character Ctrl-H
Dieser Befehl löscht das Zeichen links vom Cursor.

Move to next Line Ctrl-M
Diese Tastenkombination ist äquivalent zum Drücken der Return-Taste.

Move x numbers of Characters Ctrl-F oder Ctrl-B
Die Funktion des Befehls Ctrl-F entspricht der Taste "Cursor nach rechts", die der Tastenkombination Ctrl-B der Taste "Cursor nach links".

3.7.4 Eigene Voreinstellungen für MEmacs

Beim Starten von MEmacs wird im aktuellen Verzeichnis nach der Datei "Emacs_pro" gesucht. Wird eine Datei mit diesem Namen gefunden, so werden die darin enthaltenen MEmacs-Befehle bei jedem Neustart des Editors automatisch ausgeführt. Somit können häufige Befehle, Befehlsfolgen oder Zeichenketten selbstständig aufgerufen werden oder Tasten damit belegt werden.

Wird MEmacs im aktuellen Verzeichnis nicht fündig, so sucht er im Verzeichnis S: nach einer entsprechenden Datei. Somit kann der Editor sehr gut an die Bedürfnisse jedes einzelnen angepaßt werden.

Natürlich ist der Editor MEmcas im Vergleich zu anderen Ascii-Text-Editoren sehr spärlich ausgestattet, trotzdem sollte man ihn nicht verachten. Die wichtigsten und am häufigsten benötigten Funktionen stellt er dem Anwender zur Verfügung, so daß jeder damit seine Textdateien erstellen kann. Gerade für die Eingabe von Script-Dateien reichen die Fähigkeiten des MEmacs vollkommen aus.

3.8 Die MountList

Der Befehl MOUNT, mit dem diverse Geräte für den Gebrauch angemeldet werden können, benötigt zu den einzelnen Geräten verschiedenen Informationen, die er in der Datei "MountList" des Ordners "DEVS:" sucht und hoffentlich auch findet. Es kann sein, daß beim Erweitern des Systems die MountList durch einen Eintrag für das neue Gerät erweitert werden muß, damit der Amiga mit diesem neuen Gerät korrekt zusammenarbeiten kann. Alternativ gibt es auch bei einigen Geräten die Möglichkeit, durch das Kopieren der Treibersoftware in das Verzeichnis "Expansion" eine Erweiterung der MountList zu umgehen.

Der Eintrag der Informationen über ein Gerät ist jedoch noch keine Anmeldung dieses Gerätes. Ein Gerät muß erst mit dem Befehl "MOUNT

<Gerät>:" benutzbar gemacht werden. Beispiele hierfür sind etwa die Geräte RAD:, SPEAK: oder PIPE:. Es gibt aber auch die Möglichkeit, den Eintrag eines Gerätes als selbstständige Datei unter dem Namen des Gerätes in der Schublade "DEVS:DOSDrivers" abzulegen. Damit wird automatisch bei der Ausführung der Startup-Sequence das entsprechende Gerät angemeldet. Aus diesem Grund ist es häufig der Fall, daß keine Datei MountList mehr existiert, sondern diese in Ihre Bestandteile zerlegt als einzelnen Dateien in der Schublade "DEVS:DOSDrivers" (oder Storage/DOSDrivers) untergebracht wurde. Der AmigaDOS-Befehl MOUNT durchsucht die entsprechende Datei nach den notwendigen Einträgen. Die Eintragungen werden durch besondere Kennzeichen und Schlüsselwörter voneinander unterschieden. Dabei können einzelne Schlüsselwörter nur bei bestimmten Geräten (etwa Dateisysteme oder Handler) verwendet werden, und bei anderen ohne Funktion bleiben. Es gibt auch voreingestellte Werte, falls ein Schlüsselwort weggelassen wird, der Wert für das entsprechende Gerät jedoch benutzt werden muß.

Normalerweise braucht sich ein Anwender mit der Art und den Regeln der Einträge in der MountList nicht zu beschäftigen, da beim Erwerb einer Erweiterung die notwendigen Einträge bereits mitgeliefert werden sollten. Daher ist dieses Kapitel eher als Zusatzinformation zu bewerten, der Otto-Normal-Verbraucher wird diese Einträge wohl nie selbst zu erstellen haben.

Trotzdem wollen wir uns nun einmal die Spielregeln für die MountList näher anschauen:

- * Jeder Eintrag in der Datei MountList muß mit dem Namen des Gerätes beginnen.
- * Nach jedem Schlüsselwort muß unmittelbar das Gleichheitszeichen "=" folgen.
- * Schlüsselwörter können durch ein Semikolon voneinander getrennt werden, oder man schreibt jedes Schlüsselwort in eine neue Zeile (die zweite Methode ist sehr zu empfehlen!).
- * Jeder Eintrag endet mit dem Doppelkreuz "#", dieses Zeichen muß alleine in einer eigenen Zeile stehen.
- * Kommentare werden durch "/*" eingeleitet und durch "*/" beendet, wie in der Programmiersprache "C".

Der Befehl MOUNT kennt folgende Schlüsselwörter:

Baud= Hier wird die Baud-Rate (Übertragungsgeschwindigkeit in Bits pro Sekunde) für die serielle Schnittstelle angegeben; Hierzu ist nur sehr wenig bekannt.

Control= Hier wird die Wortlänge, Parität und Stopbit angegeben, auch hierzu gibt es wenig weitere Informationen.

BlocksPerTrack= Bei Disketten (und ähnlichen Systemen) wird die Anzahl der Sektoren auf einer Spur einer Oberfläche der bereits formatierten Diskette angegeben.

BootPri= Hiermit kann die Priorität für das Booten im Verhältnis zu den anderen bootfähigen Geräten angegeben werden. Es sind Werte zwischen -129 (nicht booten) und +127 (nur von diesem Gerät booten) möglich. Voraussetzung ist, daß das entsprechende Gerät bootfähig ist, und trotzdem angemeldet werden kann bzw. muß.

Buffers= Anzahl der Cachespeicher, die dem Gerät bei der Anmeldung zugewiesen werden. Dieser Wert kann nachträglich durch den AmigaDOS-Befehl ADDBUFFERS geändert werden. Dieser Eintrag kann nur bei Disketten und vergleichbaren Geräten verwendet werden.

BufMemType= Hier können 6 verschiedene Zahlen eingegeben werden, die Mount sagen, in welchem Speicherbereich der Cache-Puffer eingerichtet werden soll. Dabei bedeuten:

- | | |
|---------|--|
| 1 und 2 | jeder beliebige Speicherbereich im RAM |
| 3 und 4 | der Cache muß im CHIP-RAM sein |
| 5 und 6 | der Cache muß im FAST-RAM liegen. |

Dieser Eintrag kann nur bei Disketten und vergleichbaren Geräten verwendet werden.

Device= Nach diesem Schlüsselwort ist der Name des ".device"-Gerätetreibers einzugeben. Das Device muß sich im

Verzeichnis DEVS: befinden, andernfalls muß zugleich der komplette Pfadname angegeben werden.

DosType= Dieses Schlüsselwort kann nur bei Disketten und vergleichbaren Datenträgern benutzt werden und gibt verschlüsselt die Art des Dateisystems an. Wird bei diesem Gerät das neue FFS ("Fast File System") verwendet, so steht hier der Eintrag 0x444F5301, beim OFS ("Old File System") der Wert 0x444F5300. Wird kein Eintrag DosType vorgenommen, so wird automatisch das OFS verwendet.

Flags= Die hier angegebene Zahl wird beim Aufruf von "Open-Device" aus der "exec.library" verwendet, üblicherweise wird hier 0 übergeben.

Forceload= Bei einem Wert von 1 wird MOUNT gezwungen, beim Anmelden des Gerätes die Treibersoftware von der Diskette zu laden, selbst wenn schon ein Treiber in der Liste vorhanden ist. Wird hier 0 angegeben, so wird die Software nur dann geladen, wenn noch kein Eintrag in der Liste gefunden wird.

GlobVec= Hier taucht wieder der mysteriöse globale Vektor auf, ein Andenken an die alten Zeiten, in der die Programmiersprache BPCL modern war (davon stammt dieser GlobalVector ab). Es sind die drei Werte möglich, das Schlüsselwort braucht aber eigentlich nicht mehr beachtet werden.

- 1 Kein globaler Vektor
- 0 ein eigener globaler Vektor
- 1 benutze den globalen Vektor anderer.

Handler= Nach diesem Schlüsselwort ist der Name des Gerätehandler einzugeben. Der Handler muß sich im Verzeichnis L: befinden, andernfalls muß zugleich der komplette Pfadname angegeben werden.

HighCyl= Die höchstmögliche Nummer der Spuren einer Diskette oder eines vergleichbaren Datenträgers, bei anderen Geräten kann dieses Schlüsselwort nicht verwendet werden. Bei Disketten

werden normalerweise folgende Werte benutzt:

bei 880KByte-3,5"-Disketten 79

bei 5,25"-Disketten 79 oder 39

bei Festplatten je nach Größe der Festplatte

RAD: beliebig

Interleave= Dieser Wert kann nur bei Disketten und Festplatten verwendet werden und hängt einzig und allein vom Typ des angeschlossenen Gerätes ab. Dieses Schlüsselwort sollte nicht benutzt werden.

LowCyl= Auch dieses Schlüsselwort macht nur bei Festplatten und Disketten Sinn, es ist hier die kleinstmögliche Nummer aller verfügbarer Spuren einzugeben.

Mask= Bei Verwendung des FFS kommt dieses Schlüsselwort zum Einsatz, hinter dem eine Maske für die Adressen des Speicherbereichs angegeben wird, der beim DMA ("Direct Memory Access") verwendet wird. Dieses Schlüsselwort sollte nur von Fachleuten verwendet werden.

MaxTransfer= Die maximale Anzahl der Blöcke, die mit einem einzigen DMA übertragen werden können, ist hinter diesem Schlüsselwort anzugeben. Dieser Wert wird nur bei FFS und ausschließlich bei Diskette und vergleichbaren Datenträgern verwendet.

Mount= Wird hinter diesem Schlüsselwort eine positive Zahl angegeben, so lädt MOUNT die Treibersoftware für das entsprechende Gerät schon beim Anmelden, bei einer negativen Zahl erst bei der erstmaligen Benutzung.

PreAlloc= Ist der Eintrag für eine Festplatte bestimmt, so kann mit diesem Wert festgelegt werden, wieviele Sektoren beim Formatieren reserviert werden sollen, damit sie nicht für den allgemeinen Gebrauch verwendet werden. Dieser Wert hängt von der jeweils angeschlossenen Festplatte ab - die Dokumentation dazu sollte mehr Informationen anbieten.

- Priority=** Hier kann die Priorität der Gerätetreiber innerhalb aller Exec-Tasks angegeben werden. Dabei sollten bei Handlern Werte von 5 und bei Disktreibern Werte von 10 verwendet werden. Experimente mit diesen Werten sollten vermieden werden, wenn die Arbeitsweise des Exec und des Multitasking auf dem Amiga nicht detailliert bekannt ist.
- Reserved=** Nur bei Disketten und vergleichbaren Datenträgern ist dieses Schlüsselwort zu verwenden, dem ein Wert zugewiesen werden kann, der angibt, wieviele Sektoren für den Bootblock bereitgestellt werden sollen. Normalerweise wird hier die Zahl 2 eingegeben.
- Stacksize=** Hiermit kann die Größe des Stapelspeichers eingegeben werden, über den die Treibersoftware verfügen kann. Ein zu niedriger Wert kann das Betriebssystem unangenehm beeinflussen, ein Absturz ist die Folge, darum sollten Sie mit diesem Wert nicht experimentieren.
- Startup=** Hier kann eine Zeichenkette eingegeben werden, die dem Gerät, Dateisystem oder Handler beim Start als BPTR auf einen BSTR übergeben wird. Die Abkürzungen stammen ebenfalls von der Programmiersprache BPCL ab und verdienen keine nähere Beachtung.
- Surfaces=** Hiermit wird die Anzahl der Oberflächen (oder der Schreib-/Leseköpfe) eingegeben, die eine Diskette oder eine Festplatte besitzt. Bei Disketten sind dies 2, bei Festplatten können es mehr sein. Der Wert sollte nicht eigenmächtig verändert werden.
- Unit=** Bei Diskettenlaufwerken kann die Nummer angegeben werden.

3.9 AmigaDOS-Fehlermeldungen

Beim Arbeiten mit dem Amiga und dem AmigaDOS sind natürlich auch Fehler an der Tagesordnung. Die einzelnen AmigaDOS-Befehle verschlüsseln dabei jeweils den aufgetretenen Fehler in Nummern, die wiederum durch den Befehl "FAULT" interpretiert und in Klartext übersetzt werden.

Damit die Analyse des aufgetretenen Fehlers erleichtert wird, sind im folgenden alle möglichen AmigaDOS-Fehler in der Reihenfolge ihrer Fehlernummer zusammen mit der deutschen und englischen Fehlermeldung aufgeführt.

Es werden die möglichen Ursachen und Vorschläge zur Behebung des Fehlers angegeben. Es ist jedoch nicht möglich, auf alle denkbaren Fehlerquellen einzugehen, wir müssen uns auf die wahrscheinlichste Möglichkeit beschränken.

Fault 103: Speicherplatzmangel

Fault 103: not enough memory available

Ursache: Zur Abarbeitung eines Programmes (oder Befehls) reicht der verfügbare Arbeitsspeicher nicht aus, der Befehl kann deswegen nicht ausgeführt werden.

Abhilfe: Alle nicht benötigten Fenster schließen, unbenutzte Programme beenden und den Befehl neu eingeben. Wenn dies nichts nützt, nach einem Reset neu starten (es könnte zwar ausreichend Speicher vorhanden jedoch zu sehr zerstückelt sein).

Fault 105: Prozeßtabelle ist voll

Fault 105: process table full

Ursache: Die Anzahl der möglichen Prozesse innerhalb von AmigaDOS wurde überschritten. Vor der Version 2.0 war die Anzahl der Prozesse auf 20 begrenzt.

Abhilfe: Einen oder mehrere Prozesse beenden.

Fault 114: Falsches Namensmuster

Fault 114: bad template

Ursache: Dieser Fehler ist dem Programmierer oder Anwender zuzuschreiben, der Befehl wurde mit einem ungültigen Namensmuster als Argument gestartet, das der Befehl nicht auswerten kann.

Abhilfe: Syntax des Befehl und der Argumente überprüfen und korrigieren.

Fault 115: Ungültiger Zahlenwert

Fault 115: bad number

Ursache: Es wurde ein Argument eingegeben, das keine Zahl ist, obwohl der Befehl an einer Stelle eine Zahl erwartet. Häufig ist die Ursache ein normaler Tippfehler.

Abhilfe: Syntax des Befehl und der Argumente überprüfen und korrigieren.

Fault 116: Gefordertes Argument fehlt

Fault 116: required argument missing

Ursache: Ein notwendiges Argument wurde nicht eingegeben. Es gibt Befehle, die unbedingt ein Argument benötigen, um arbeiten zu können.

Abhilfe: Syntax des Befehl und der Argumente überprüfen und korrigieren.

Fault 117: Argument nach Schlüsselwort fehlt

Fault 117: value after keyword missing

Ursache: Nach einem Schlüsselwort wird vom Befehl ein entsprechender Wert erwartet, aber nicht eingegeben.

Abhilfe: Syntax des Befehl und der Argumente überprüfen und korrigieren.

PAG Sandini

Fault 118: Falsche Anzahl an Argumenten**Fault 118: wrong number of arguments**

Ursache: Es wurden zu viele Argumente übergeben, mit denen der Befehl nichts anfangen kann. Oft wurde bei einem Pfadname statt des “/” ein Leerzeichen eingegeben, oder (vor allem in älteren Versionen) eine Zeichenkette nicht durch Anführungszeichen eingeschlossen.

Abhilfe: Syntax des Befehl und der Argumente überprüfen und korrigieren.

Fault 119: Ungerade Anzahl von Anführungszeichen**Fault 119: unmatched quotes**

Ursache: Es wurden eine ungerade Anzahl von Anführungszeichen eingegeben, so daß dem letzten kein schließender Partner zugewiesen werden konnte. Möglicherweise wurde vergessen, einem Anführungszeichen, daß in den Text übernommen werden soll, einen Stern voranzustellen.

Abhilfe: Syntax des Befehl und der Argumente überprüfen und korrigieren.

Fault 120: Argumentzeile ist ungültig oder zu lang**Fault 120: argument line invalid or too long**

Ursache: Die Befehlszeile enthält zu viele Zeichen oder kann aus anderen Gründen nicht analysiert werden.

Abhilfe: Syntax des Befehl und der Argumente überprüfen und korrigieren.

Fault 121: Datei ist nicht ausführbar**Fault 121: file is not executable**

Ursache: Sie haben versucht, eine Datei als Programm zu starten, obwohl diese Datei keinen ausführbaren Programmcode enthält. Sollte

es sich bei der Datei um eine Befehlsdatei handeln, so ist sie entweder durch den Befehl EXECUTE zu starten oder mit dem Schutzbit "s" (siehe AmigaDOS-Befehl PROTECT) zu versehen.

Abhilfe: Überprüfen Sie, ob die entsprechende Datei eine Befehlsdatei ist und starten Sie diese dann mit EXECUTE.

Fault 122: Ungültige residente Library

Fault 122: invalid resident library

Ursache: Diese neu eingeführte Fehlermeldung sollte eigentlich nie erscheinen. Hier wird Ihnen mitgeteilt, daß Sie einen Befehl starten wollten, der nur mit den älteren Libraries arbeitet, bei den neuen jedoch nicht mehr. (Nach den Richtlinien von Commodore sollten jedoch alle Libraries abwärtskompatibel sein).

Abhilfe: Versuchen Sie, einen gleichwertigen Befehl für das neue Betriebssystem zu verwenden.

Fault 202: Objekt ist in Gebrauch

Fault 202: object is in use

Ursache: Sie haben vermutlich versucht, eine Datei oder ein Verzeichnis zu löschen, daß derzeit noch durch andere Programme oder Prozesse verwendet wird. Es kann besonders bei Verzeichnissen vorkommen, daß diese nicht gelöscht werden können, weil sie in einem Suchpfad aufgenommen wurden, oder ein logisches Gerät zugewiesen bekommen haben.

Abhilfe: Datei oder Verzeichnis nicht löschen, oder versuchen, die oben genannten Sperren aufzuheben.

Fault 203: Objekt existiert bereits

Fault 203: object already exists

Ursache: Beim Erstellen eines neuen Verzeichnisses oder beim Umbenennen von Dateien oder Verzeichnissen wurde ein Name

verwendet, der bereits im entsprechenden Verzeichnis durch eine andere Datei oder ein anderes Verzeichnis belegt ist. Jede Datei und jedes Unterverzeichnis in einer Schublade muß einen einzigartigen Namen besitzen.

Abhilfe: Verwenden Sie einen anderen Namen.

Fault 204: Verzeichnis nicht gefunden

Fault 204: directory not found

Ursache: Es wurde bei einem Befehl ein Verzeichnis angegeben, das nicht existiert (normalerweise liegt ein Tippfehler vor).

Abhilfe: Eingabe überprüfen und korrigieren.

Fault 205: Objekt nicht gefunden

Fault 205: object not found

Ursache: Es existiert keine Datei mit dem eingegebenen Namen, auch hier handelt es sich in der Regel um einen einfachen Tippfehler.

Abhilfe: Eingabe überprüfen und korrigieren.

Fault 206: Ungültige Fensterparameter

Fault 206: invalid window description

Ursache: Dieser Fehler tritt bei ungültigen Werten für die Gestalt von Konsolen-Fenster wie RAW: und CON: auf. Es können speziell folgende Angaben falsch sein:

- * Die X- und Y-Koordinaten des Fenster können außerhalb des darstellbaren Bereichs liegen.
- * Das Fenster kann zu groß sein, oder negative Werte für Höhe oder Breite besitzen.
- * Der Amiga stellt den Bildschirm im PAL-Modus dar, bei dem es in einigen Fat Agnus-Chips zu diesem Fehler kommen kann.

Abhilfe: Die Fensterparameter überprüfen und korrigieren.

Fault 207: Objekt ist zu groß

Fault 207: object too large

Ursache: Ich konnte diesen Fehler bislang noch nicht erzeugen, so daß ich auch keine Angaben zur Ursache machen kann (in der offiziellen AmigaDOS-Dokumentation wurde dieser Fehler nicht einmal erwähnt).

Abhilfe: Wo kein Fehler auftritt, braucht man auch keine Abhilfe, hoffentlich bleibt es dabei!

Fault 209: Unbekannter DOS-Packet-Request-Typ

Fault 209: packet request type unknown

Ursache: Dieser Fehler tritt auf, wenn die Meldungen innerhalb von AmigaDOS durcheinander geraten, etwa von einem Gerätetreiber unmögliche Aktionen erwartet werden oder ähnliches. Auch dieser Fehler sollte bei veröffentlichter Software nicht auftauchen.

Abhilfe: Versuchen Sie, einen fehlerfreien Gerätetreiber zu erhalten, normalerweise muß die Vertreiberfirma dafür sorgen.

Fault 210: Ungültiger Objektname

Fault 210: object name invalid

Ursache: Eine Befehlszeile oder ein Dateiname enthält ungültige Zeichen, wie etwa Steuerzeichen oder der Name einer Datei oder eines Verzeichnisses ist zu lang.

Abhilfe: Am besten geben Sie diesen Befehl oder den Namen noch einmal ein.

Fault 211: Ungültiger Lock-Zugriff auf Objekt

Fault 211: invalid object lock

Ursache: Ein Programm hat versucht, eine Datei oder ein Verzeichnis für sich zu reservieren (LOCK), obwohl das entsprechende

AmigaDOS-Objekt nicht existiert oder nicht reserviert werden kann. Dieser Fehler sollte ebenfalls in veröffentlichter Software nicht auftauchen.

Abhilfe: Eingaben überprüfen und gegebenenfalls korrigieren. Sollte es sich um kommerzielle Software handeln, setzen Sie sich am besten mit der Firma in Verbindung.

Fault 212: Objekt ist nicht vom geforderten Typ

Fault 212: object is not of required type

Ursache: Sie haben vermutlich versucht, etwas mit einer Datei, einem Verzeichnis oder einem Gerät zu machen, was grundsätzlich mit einem AmigaDOS-Objekt dieses Typs nicht gemacht werden kann.

Abhilfe: Befehlssyntax und Argumente überprüfen und korrigieren.

PAG Sandini

Fault 213: Disk ist nicht gültig

Fault 213: disk not validated

Ursache: Dieser Fehler wird in der Regel von kaputten Disketten hervorgerufen. Es kann aber auch sein, daß bereits versucht wurde, auf die Daten der Diskette zuzugreifen, obwohl die systembedingte Überprüfung einer Diskette unmittelbar nach deren einlegen in ein Laufwerk noch nicht abgeschlossen war.

Abhilfe: Warten, ob der Fehler nach einer Pause erneut auftritt. Ist dies der Fall, so ist die Diskette kaputt und kann eventuell mit dem AmigaDOS-Befehl DISKDOCTOR gerettet oder aber (nicht immer) durch formatieren wieder benutzbar gemacht werden.

Fault 214: Disk ist schreibgeschützt

Fault 214: disk is write-protected

Ursache: AmigaDOS hat den Auftrag bekommen, auf eine schreibgeschützte Diskette zu schreiben wodurch dieser Fehler entsteht.

Möglicherweise haben Sie das falsche Diskettenlaufwerk als Ziel angegeben.

Abhilfe: Schreibschutz der Diskette entfernen. Vorsicht: Ein Schreibschutz soll vor dem Beschreiben schützen - es gibt immer eine ganze Reihe von Gründen, warum auf eine Diskette nicht geschrieben werden soll. Im Zweifelsfall sollte eine andere Diskette beschrieben werden.

Fault 215: Umbenennen auf anderen Datenträger versucht

Fault 215: rename across devices attempted

Ursache: Es wurde versucht, mit dem Befehl RENAME eine Datei von einem Gerät auf ein anderes zu verschieben. Dies kann möglicherweise auf einen Tippfehler zurückzuführen sein.

Abhilfe: Handelte es sich um einen Tippfehler, muß der Befehl noch einmal korrekt eingegeben werden, sollte die Datei tatsächlich auf ein anderes Gerät gebracht werden, muß der Befehl COPY verwendet werden.

Fault 216: Verzeichnis ist nicht leer

Fault 216: directory not empty

Ursache: Es wurde versucht, ein Verzeichnis zu löschen, in dem noch weitere Unterverzeichnisse oder Dateien vorhanden sind.

Abhilfe: Entweder erst alle anderen Unterverzeichnisse und Dateien löschen, bevor das Verzeichnis selbst entfernt wird, oder (VORSICHTIG!) mit dem Schlüsselwort ALL alle Daten in diesem Verzeichnis eliminieren.

Fault 217: Zu tiefe Schachtelung

Fault 217: too many levels

Ursache: Die maximale Anzahl von Softlinks (15) wurde überschritten.

Abhilfe: Die entsprechende Anzahl verringern oder auf einen neuen Link verzichten.

Fault 218: Gerät (oder Datenträger) ist nicht angemeldet

Fault 218: device (or volume) is not mounted

Ursache: Sie haben versucht, ein Gerät anzusprechen, das AmigaDOS nicht kennt. Vorher erschien vermutlich der Requester "Please insert Disk xxx: in any Drive", der mit "Cancel" beantwortet wurde. Der Fehler kann auf eine falsche Schreibweise zurückgeführt werden, oder es wurde vergessen, daß entsprechende Gerät mit dem Befehl MOUNT anzumelden.

Abhilfe: Eingabe überprüfen und korrigieren oder Gerät anmelden.

Fault 219: Fehler bei Suchlesen

Fault 219: seek failure

Ursache: AmigaDOS meldet, daß bei einem Gerätetreiber versucht wurde, die Funktion Seek der "dos.library" oder des Device selbst aufzurufen, obwohl der Treiber keine derartige Funktion unterstützt. Oder der Treiber unterstützt diese Funktion, hat aber beim Funktionsaufruf ungültige Werte übergeben bekommen.

Abhilfe: Dieser Fehler tritt normalerweise nur bei der Programmierung auf und sollte bei kommerzielle Software nicht erscheinen. Gegebenenfalls muß die Vertriebs-Firma informiert werden.

Fault 220: Kommentar ist zu lang

Fault 220: comment is too long

Ursache: Sie haben versucht, einer Datei einen Kommentar mit mehr als 79 Zeichen zu verpassen.

Abhilfe: Den Kommentar kürzen (siehe dazu auch den AmigaDOS-Befehl FILENOTE).

Fault 221: Disk ist voll

Fault 221: disk is full

Ursache: Beim Schreiben von Daten auf einen Datenträger (Diskette, Festplatte oder RAM-Disk) wurde die Speicherkapazität des Mediums überschritten. Wurden einzelnen Dateien mit dem Befehl COPY unter Verwendung von Jokerzeichen kopiert, so können auch weniger Dateien mit gezielteren Namensmuster oder explizierter Angabe der Dateinamen kopiert werden.

Abhilfe: Weniger auf den Datenträger schreiben, eventuell neuen Datenträger verwenden.

Fault 222: Objekt ist löschgeschützt

Fault 222: object is protected from deletion

Ursache: Es wurde versucht, eine Datei oder ein Verzeichnis zu löschen, dessen Schutzbit "D" (deleteable) nicht gesetzt ist.

Abhilfe: Objekt nicht löschen, das D-Bit mit dem AmigaDOS-Befehl PROTECT setzen oder beim Befehl DELETE das Schlüsselwort FORCE verwenden.

Fault 223: Datei ist schreibgeschützt

Fault 223: file is write protected

Ursache: Es wurde versucht, in eine Datei zu schreiben, die durch das gelöschte Schutzbit "W" (writeable) vor Veränderungen geschützt ist. Dieses Schutzbit existierte zwar schon vor der Version 2.0, wird jedoch erst seit der Version 2.0 beachtet.

Abhilfe: Nicht in die Datei schreiben oder mit dem Befehl PROTECT das Schutzbit "W" setzen.

Fault 224: Datei ist lesegeschützt

Fault 224: file is read protected

Ursache: Es wurde versucht, Daten aus einer Datei zu lesen, die durch das gelöschte Schutzbit "R" (readable) davor bewahrt wurde.

Dieses Schutzbit existierte zwar schon vor der Version 2.0 wird jedoch erst seit der Version 2.0 beachtet.

Abhilfe: Die Datei nicht lesen oder mit dem Befehl PROTECT das Schutzbit "R" setzen.

Fault 225: Keine gültige DOS-Disk

Fault 225: not a valid DOS disk

Ursache: Die angesprochene Diskette wurde nicht im AmigaDOS-Format formatiert oder wurde inzwischen zerstört. Möglicherweise wurde die Diskette mit dem Befehl DISKCOPY nur teilweise kopiert.

Abhilfe: Diskette formatieren oder andere Diskette verwenden.

Fault 226: Keine Diskette im Laufwerk

Fault 226: no disk in drive

Ursache: Im angegebenen Laufwerk befindet sich keine Diskette.

Abhilfe: Diskette einlegen.

Fault 232: Keine weiteren Verzeichniseinträge

Fault 232: no more entries in directory

Ursache: Dieser Fehler sollte bei der normalen Arbeit mit einer Shell nicht auftreten, da er durch den Aufruf der Funktion "ExNext" der "dos.library" entsteht, wenn versucht wird, in einem Verzeichnis weitere Dateien zu untersuchen, wenn bereits alle behandelt wurden.

Abhilfe: Korrekt programmieren; in kommerziellen Programmen dürfte dieser Fehler nicht auftreten, gegebenenfalls muß sich der Anwender mit der Vertriebsfirma in Verbindung setzen.

Fault 233: Objekt im Verbund**Fault 233: object is soft link**

Ursache: Sie haben versucht, einen Befehl auf ein AmigaDOS-Objekt anzuwenden, das eigentlich nur eine Verbindung (Link) zu dem eigentlichen Objekt darstellt. Einige Befehle sind nur auf die Objekte direkt, nicht aber auf die Soft-Links anwendbar.

Abhilfe: AmigaDOS-Befehl direkt auf das eigentliche Objekt anwenden.

Fault 234: Verbundobjekt**Fault 234: object is linked**

Ursache: Dieser Fehler ist wie Nummer 233, es handelt sich hierbei lediglich um einen sogenannten Hard-Link.

Abhilfe: Siehe Fehler 233.

Fault 235: Ungültiger Hunk in zu ladender Datei**Fault 235: bad loadfile hunk**

Ursache: Es wurde versucht, ein Programm zu starten, das Teile besitzt, die von AmigaDOS nicht anerkannt werden. Dieser Fehler ist bei mir beim Versuch aufgetreten, ein Programm zu laden, das für ältere Amigas entwickelt wurde.

Abhilfe: Neuere Versionen des Programms besorgen.

Fault 236: Funktion ist nicht implementiert**Fault 236: function not implemented**

Ursache: Vermutlich tritt dieser Fehler beim Versuch auf, eine Routine der dos.library aufzurufen, die bei neueren Versionen nicht mehr vorhanden ist oder aber in späteren Versionen hinzugefügt werden soll. Dieser Fehler sollte bei kommerziellen Programmen nicht auftreten.

Abhilfe: Beim Entwickler beschweren!

Die folgenden vier Fehler werden nur der Vollständigkeit halber aufgeführt, sie treten bei der Verwendung der AmigaDOS-Befehle nicht auf. Seit der Version 2.04 wird es mehreren Programmen ermöglicht, gleichzeitig auf Teile der gleichen Datei zurückzugreifen. Diese Teile werden als "Records" bezeichnet.

Fault 240: Datensatz nicht gesperrt**Fault 240: record not locked**

Ursache: Es wird versucht, auf einen Teil einer Datei zuzugreifen, ohne den Teil vorher für diesen Zweck reserviert zu haben.

Fault 241: Kollision bei Datensatzsperre**Fault 241: record lock collision**

Ursache: Es wird der gleiche Schlüssel von zwei verschiedenen Programmen gleichzeitig verwendet. Es ist zwar möglich, auch den gleichen Teil von verschiedenen Programmen aus gleichzeitig zu benutzen, allerdings muß in diesem Fall jedes Programm selbst den Teil reserviert haben.

Fault 242: Zeitüberschreitung bei Datensatzsperre**Fault 242: record lock timeout**

Ursache: Es gibt Schlüssel, die nur eine gewisse Zeit lang benutzt werden können. Wird ein Schlüssel auch nach dieser Zeitspanne benutzt, erscheint diese Fehlermeldung.

Fault 243: Fehler bei Datensatzfreigabe**Fault 243: record unlock error**

Ursache: Dieser Fehler tritt dann auf, wenn versucht wird, einen Schlüssel zurückzugeben, obwohl nie ein solcher geholt wurde.

Fault 303: Pufferüberlauf**Fault 303: buffer overflow**

Ursache: Bei der Ausführung eines Befehls reichte der vorhandene Pufferspeicher nicht aus. Möglicherweise ist der gesammte verfügbare Speicher zu knapp.

Abhilfe: Bei dem Befehl COPY den Pufferspeicher durch das Argument BUFFER vergrößern. Gegebenenfalls können Fenster geschlossen und unbenutzte Programme beendet werden.

Fault 304: * Abbruch****Fault 304: ***Break**

Ursache: Dieser Fehler ist eigentlich kein Fehler, sondern lediglich die Meldung, daß ein Programm oder eine Befehlsdatei mit dem AmigaDOS-Befehl BREAK oder der entsprechenden Tastenkombination Ctrl-C bzw. Ctrl-D abgebrochen wurde.

Fault 305: Datei ist nicht ausführbar**Fault 305: file not executable**

Ursache: Es wurde der Name einer Datei als Befehl eingegeben, obwohl diese Datei keinen ausführbaren Programmcode besitzt. Handelt es sich dabei um eine Befehlsdatei, so wurde das Schutzbit "S" nicht gesetzt.

Abhilfe: Anderen Befehl eingeben. Handelt es sich um ein Script, so kann mit dem AmigaDOS-Befehl PROTECT das Schutzbit "S" gesetzt werden oder die Befehlsdatei mit dem Befehl EXECUTE gestartet werden.

4 - System-Fehlermeldungen (Guru-Meditations)

Dieses hier aufgeführten System-Fehlermeldungen beziehen sich auf die Nummern, die in den Guru-Meditationen oder in den Requestern "Software Failure..." vorkommen. Der Anwender kann gegen das Auftreten dieser Fehler eigentlich sehr wenig machen - außer das fehlerhafte Programm nicht mehr benutzen.

Die Fehler entstehen durch unkorrekte Programme, die nur der Programmierer selbst verbessern kann, der Anwender kann darauf keinen Einfluß nehmen. Daher ist eigentlich die Angabe dieser Fehlermeldungen nur für diejenigen unter Ihnen notwendig, die selbst Programme entwickeln, da aber erfahrungsgemäß auch alle anderen wissen wollen, was denn nun für ein Fehler aufgetreten ist, folgt nun die Liste der Guru-Meditations - jedoch ohne Vorschläge zur Abhilfe.

PAG Sandini

4.1 - Der Aufbau der Fehlernummer

In einer Guru-Meditation erscheinen zwei Zahlen mit jeweils acht Ziffern. Dabei stellen die letzten acht Ziffern lediglich die Adresse des Tasks dar, der den Fehler hervorgerufen hat. Die ersten acht Ziffern teilen sich nach folgenden Schema auf:

AABBCCCC

Die ersten beiden Ziffern (A) bezeichnen das sogenannte Subsystem, das den Fehler verursachte, die folgenden beiden (B) geben allgemeinere Informationen zum aufgetretenen Fehler und die letzten vier (C) charakterisieren schließlich den Fehler im Detail. Die Fehlernummern werden in Hexadezimaler Schreibweise angegeben, so daß darauf verwiesen werden muß, daß der obige Fehlercode ein Schema darstellt - keine Zahlen, gleiches gilt für die Muster der Fehlermeldungen, die nach der Liste der Prozessor-Fehler aufgeführt ist. Generell kann man folgendes vereinbaren: Tauchen im

Fehlercode die Buchstaben der Gruppen AA, BB oder CCCC auf, so sind diese zu ersetzen, handelt es sich bereits um eine Hexadezimalzahl, so sind niemals zwei gleiche Buchstaben nebeneinander.

In der Guru-Meditation (mit dem schönen rot/orange blinkenden Kasten) fordert der Amiga den Benutzer auf, die linke Maustaste zu drücken. Mit der rechten Maustaste sollte eigentlich in bestimmten Fällen eine Rückkehr in den normalen Betrieb des Amigas ermöglicht werden. Denn wenn die erste Ziffer kleiner als 8 ist, sollte eine Rückkehr möglich sein, nur bei einer Ziffer größer oder gleich 8 ist die Rückkehr mit Sicherheit nicht möglich. Leider war hier bei Commodores Entwicklern der Wunsch Vater des Gedankens, denn in der Realität hat sich gezeigt, daß auch bei fast allen sogenannten "recoverable alerts" (Gurus mit Rückkehrmöglichkeit) ein Totalreset des Amigas durchgeführt wurde, selbst wenn die rechte Maustaste gedrückt wurde.

Seit der Version 2.0 klappt zumindest der Stopp eines fehlerhaften Tasks, wenn der Requester "Software Failure" erscheint. Früher konnte auch nicht nur der fehlerhafte Task gestoppt werden, sondern auch hier wurde automatisch ein Reset ausgelöst.

4.2 - Prozessor-Fehler

System-Fehler, deren ersten beiden Ziffern eine 0 enthalten, besitzen eine Ausnahmestellung unter allen Fehlercodes, da diese Fehler vom Hauptprozessor (CPU) selbst festgestellt und gemeldet werden - alle anderen werden vom Betriebssystem des Amigas ausgegeben.

Dem Prozessor 68000 stehen für die Behandlung der aufgetretenden Fehler insgesamt 256 Vektoren (Sprungadressen) zur Verfügung, von denen 64 bereits durch die CPU selbst festgelegt sind, die restlichen 192 Vektoren können vom Benutzer (dazu zählt hier auch das Betriebssystem) frei definiert werden.

Hier die nun die Liste aller Prozessor-Fehler:

Code	Name	Ursache
0000 0002	Bus Error	Ein angeschlossenes Gerät hält sich nicht an die Taktrate des Adress- oder Datenbus.
0000 0003	Adress Error	Ein Programm verwendet eine falsche Adresse (meist unzulässigerweise ungerade Adresse).
0000 0004	Illegal Instruction	Bei der Abarbeitung eines Programmes trat ein Befehl auf, den der 68000 nicht kennt.
0000 0005	Devide by Zero	Ein Programm hat versucht, durch Null zu dividieren.
0000 0006	CHK Instruction	Bei einer Überprüfung eines Registerinhalts wurden Grenzen überschritten.
0000 0007	TRAPV	TRAPV springt auf einen Trap-Vektor, wenn das Instruction V-Flag gesetzt ist.
0000 0008	Privilege	Es taucht im User-Modus des Prozessors ein Violation Befehl auf, der nur im Supervisor-Mode ausgeführt werden kann.
0000 0009	Trace	Der Prozessor befindet sich im Einzelschritt-Modus.
0000 000A	OP Code 1010	Es wurde ein Befehl der Gruppe 1010 ausgeführt.
0000 000B	OP Code 1111	Es wurde ein Befehl der Gruppe 1111 ausgeführt.

4.3 Betriebssystem-Fehler

Alle nun folgenden Guru-Meditations werden vom Betriebssystem des Amigas erzeugt, die ersten beiden Ziffern kennzeichnen den Teil, der den Fehler auslöst. Es werden im folgenden die Muster der Fehlermeldungen gezeigt.

Fehler der Libraries:

01BB CCCC	Exec
02BB CCCC	Graphics
03BB CCCC	Layers
04BB CCCC	Intuition
05BB CCCC	Math
06BB CCCC	Clist
07BB CCCC	DOS
08BB CCCC	RAM
09BB CCCC	Icon
0ABB CCCC	Expansion

Fehler der Devices:

10BB CCCC	Audio
11BB CCCC	Console
12BB CCCC	GamePort
13BB CCCC	Keyboard
14BB CCCC	TrackDisk
15BB CCCC	Timer

Fehler durch Resource:

20BB CCCC	CIA
21BB CCCC	Disk
22BB CCCC	Misk

PAG Sandini

Sonstige Fehler:

30BB CCCC	Bootstrap
31BB CCCC	Workbench
32BB CCCC	DiskCopy

Übergeordnete Fehler:

AA00 CCCC	Der Fehler kann nicht zugeordnet werden.
AA01 CCCC	Der Fehler ist auf mangelnden Speicher zurückzuführen.
AA02 CCCC	Eine Library konnte nicht erstellt werden.
AA03 CCCC	Eine Library konnte nicht geöffnet werden.
AA04 CCCC	Ein Device konnte nicht geöffnet werden.
AA05 CCCC	(OpenResource Error) Ein Hardware-Baustein hat nicht reagiert.
AA06 CCCC	Fehler bei einer Ein- oder Ausgabe.
AA07 CCCC	Es fehlt ein Signal.

4.3.1 - Die Fehler der "exec.Library"

8100 0001	68000 Exception Vector Checksum Bei der Ausnahmebehandlung des Prozessors trat ein Fehler der Prüfsumme auf.
8100 0002	ExecBase Checksum Die Startadresse der "exec.library" hat eine falsche Prüfsumme.
8100 0003	Library Checksum Error Eine Library hat eine falsche Prüfsumme.
8100 0004	No Memory to Make Library Für eine Library reicht der Speicher nicht aus.

- | | |
|-----------|--|
| 8100 0005 | Corrupted Memory List
Die Speicherverwaltungsliste wurde zerstört. |
| 8100 0006 | No Memory For Interrupt Servers
Für eine Interruptbehandlung reicht der Speicher nicht aus. |
| 8100 0007 | initAPTR
Ein Zeiger auf eine Adresse ist falsch. |
| 8100 0008 | Semaphore Corrupt
Eine Semaphore ist zerstört. |
| 8100 0009 | Free Twice
Ein Speicherplatz wurde zweimal freigegeben. |
| 8100 000A | Bogus Exception
Es wurden reservierte Vektoren verwendet. |

PAG Sandini

4.3.2 - Fehler der "graphics.library"

- | | |
|-----------|---|
| 8201 0001 | No Memory For Copper Display List
Für eine Copperliste reicht der Speicher nicht aus. |
| 8201 0002 | No Memory For Copper Instruction List.
Für eine Copper-Befehlsliste reicht der Speicher nicht aus. |
| 8201 0003 | Copper List Overload
Die Liste für den Copper ist voll. |
| 8201 0004 | Copper Intermediate List Overload
Die Struktur der Copperliste wurde zerstört. |
| 8201 0005 | No Memory For Copper List Head
Für den Kopf der Copperliste reicht der Speicher nicht aus. |

- 8201 0006 Long Frame, No Memory
Für die Copperliste I reicht der Speicher im Interlace-Betrieb nicht aus.
- 8201 0007 Short Frame, No Memory
Für die Copperliste I reicht der Speicher im Interlace-Betrieb nicht aus.
- 8201 0008 No Memory For Flood Fill
Zur Ausführung des Fill-Befehls reicht der Speicher nicht aus.
- 8201 0009 Text, No Memory For TmpRas
Für das Anlegen einer zeitweiligen TmpRas-Datei reicht der Speicher nicht aus.
- 8201 000A No Memory For BltBitMap
Für die Blitter Bitmap reicht der Speicher nicht aus.
- 8201 000B Region Memory
Ein Speicherbereich wurde falsch angegeben.
- 8201 0030 MakeVPort
Beim Errichten eines ViewPorts trat ein Fehler auf.
- 8201 1234 GfxNoLCM
Der Zwischenspeicherbereich ist nicht frei.

4.3.3 - Fehler der "layers.library"

- 8301 0001 LayersNoMem
Für die Layers reicht der Speicherplatz nicht aus.

4.3.4 - Fehler der "intuition.library"

- | | |
|-----------|--|
| 8400 0001 | Unknown Gadget Type
Der angegebene Gadget-Typ ist nicht bekannt. |
| 0400 0001 | Wie oben, jedoch abfangbar (hoffentlich!). |
| 8401 0002 | No Memory to create Port
Für das Errichten eines neuen Ports reicht der Speicher nicht aus. |
| 8401 0003 | Item Plane Alloc, No Memory
Für die Darstellung der Menüleiste reicht der Speicher nicht aus. |
| 8401 0004 | Sub Alloc, No Memory
Für die Darstellung eines Untermenüs reicht der Speicher nicht aus. |
| 8401 0005 | Plane Alloc, No Memory
Für die Darstellung der Kopfzeile des Menüs reicht der Speicher nicht aus. |
| 8400 0006 | Item Box Top Less Than Real Zero
Die obere Grenze einer Item-Box liegt unter der absoluten Nullmarke. |
| 8401 0007 | No Memory To Open Screen
Zum Öffnen des Bildschirms reicht der Speicher nicht aus. |
| 8401 0008 | Open Screen, Raster Alloc, No Memory
Für den RastPort reicht der Speicher nicht aus. |
| 8400 0009 | Open Sys Screen, Unknown Type
Es wurde ein unbekannter Screen-Typ angegeben. |
| 8401 000A | Add SW Gadgets, No Memory
Für ein Gadget reicht der Speicher nicht aus. |

- 8401 000B No Memory to Open Window.
Zum Öffnen eines Fensters reicht der Speicher nicht aus.
- 8400 000C Bad State Return Entering Intuition
Beim Öffnen von Intuition war die Statusangabe falsch.
- 8400 000D Bad Message Received by IDCMP
Der "Intuition Direct Communication Message Port" erhielt eine ungültige Meldung.
- 8400 000E Wierd Echo Causing Incomprehension
Für den Zugriff auf die "Distant Echo List" reicht der Speicher nicht aus.
- 8400 000F Could not Open The Console Device
Das Console.Device konnte nicht geöffnet werden.

4.3.5 - Fehler der "dos.library"

- 0701 0001 No Memry At Startup
Beim Startup reicht der Speicher nicht aus.
- 0700 0002 EndTask didn't
EndTask hat nicht oder nur fehlerhaft gearbeitet.
- 0700 0003 Qpkt Failure
Bei der Übertragung eines Datenpaketes trat ein Fehler auf.
- 0700 0004 Unexpectes Packet Received
Es wurde ein Datenpaket empfangen, das nicht erwartet wurde.
- 0700 0005 Freevec Failed
Freevec hat nicht oder nur fehlerhaft gearbeitet.
- 0700 0006 Disk Block Sequence Error
Eine Disk-Block-Sequenc ist fehlerhaft.

0700 0007	Bitmap Corrupt Die Bitmap ist fehlerhaft oder zerstört.
0700 0008	Key already Free Ein Schlüssel auf eine Datei oder Verzeichnis wurde schon längst wieder freigegeben.
0700 0009	Invalid Checksum Die Prüfsumme ist nicht korrekt.
0700 000A	Disk Error Bei einem Diskettenlaufwerk trat ein Fehler auf.
0700 000B	Key Out Of Range Eine Datei-Nummer liegt außerhalb des zulässigen Bereichs.
0700 000C	Bad Overlay Der Overlay-Hunk ist nicht in Ordnung.

4.3.6 - Fehler der "ram.library"

0800 0001	Bad Segment List Die Speicherverwaltungsliste ist fehlerhaft.
-----------	--

4.3.7 - Fehler der "expansion.library"

0A00 0001	Bad Expansion Free Bei einer Hardwareerweiterung oder deren Treibersoftware trat ein Fehler auf.
-----------	--

4.3.8 - Fehler des "trackdisk.device"

- 1400 0001 Calibrate: Seek Error
Beim Suchen auf der Diskette trat ein Fehler auf.
- 1400 0002 Delay: Error On Timer Wait
Beim Warten auf einen Timer-Impuls trat ein Fehler auf.

4.3.9 - Fehler des "timer.device"

- 1500 0001 Bad Request
Ein Zugriffsversuch ist fehlgeschlagen.
- 1500 0002 Bad Supply
Durch eine un stabile Netzfrequenz wurde die Steuerung fehlerhaft.

4.3.10 - Fehler der Disk Resource

- 2100 0001 Get Unit: Alreada has Disk
Das DiskChange-Signal ist fehlerhaft.
- 2100 0002 Interrupt: No Active Unit
Es wurde versucht, ein Gerät zu unterbrechen, das gar nicht im Einsatz war.

4.3.11 - Fehler des BootStrap

- 3000 0001 Boot Code Returned an Error
Die "dos.library" konnte nicht gefunden werden.

5. ARexx für Einsteiger

In diesem Kapitel wollen wir Sie von der Pike auf mit den grundlegenden Eigenschaften der Programmiersprache ARexx vertraut machen, so daß Sie zukünftig auch eigenständig kleinere Programme schreiben können.

Natürlich kann ein Kompakt-Kurs wie dieser nicht alle Fähigkeiten der Programmiersprache ARexx aufzeigen: Wir werden uns deshalb auf das absolut notwendige beschränken, aber trotzdem versuchen, Ihnen einen weiten Überblick zu verschaffen.

Was ist ARexx?

Die Programmiersprache REXX wurde im Zeitraum von 1979 bis 1985 von dem Engländer Mike F. Cowlishaw entwickelt, "um das Programmieren zu erleichtern, in der Überzeugung, daß der Entwurf hochwertiger Programme durch einfache Handhabung der Sprache unterstützt wird". Selbst wenn REXX heute nicht die Popularität der Volk-Programmierprachen Pascal und BASIC erreicht, so vereint es doch viele der positiven Eigenschaften dieser beiden Sprachen in sich: REXX ist einfach zu erlernen, bietet aber trotzdem sehr mächtige Befehle und eine Vielzahl von Kontrollstrukturen. Da REXX eine Interpreter-Sprache ist, fallen aufwendige Compiler-Läufe bei der Programmentwicklung weg, denn Programme können eingetippt und gleich getestet werden.

Die von William Hawes portierte Version ARexx umfaßt den kompletten Befehlssatz des ursprünglichen REXX, und erweiterte ihn um viele nützliche Zusatzfunktionen.

Einsatzgebiete von ARexx

REXX war ursprünglich als universell einsetzbare Programmiersprache gedacht, mit der beispielsweise Studenten das strukturierte Programmieren erlernen sollten. Schon bald kristallisierte sich jedoch ein besonderes Einsatz-

gebiet heraus: Mit den vorhandenen Befehlen zur Zeichenkettenverarbeitung eignet sich ARexx hervorragend auch als Makro-Programmiersprache für Anwendungsprogramme verschiedenster Kategorien. Die Vorteile liegen auf der Hand: Hersteller müssen nicht erst unter beträchtlichem Aufwand eine eigene Kommandosprache "erfinden", und Anwender müssen nicht für jedes größere Programm eine neue Kommandosprache lernen, sondern jeweils nur die applikationsspezifischen Sonderbefehle.

Das Host-Konzept von ARexx

Zwischen Applikation und ARexx-Interpreter existiert eine Schnittstelle, über die sich der ARexx-Interpreter und die Applikation gegenseitig Befehle zur Ausführung übergeben können. Das Applikationsprogramm wird somit zum "Function Host", der den ARexx-Interpreter um neue Funktionen erweitert, während der Interpreter für die Applikation Makro-Programme startet, und deren Variablen und Kontrollstrukturen verwaltet. Darüber hinaus gibt es auch reine Funktionsbibliotheken, die nur Funktionen zur Verfügung stellen, aber sonst keine Ansprüche an den Interpreter stellen: Sie werden "Function Libraries" genannt.

Tatsächlich sind alle ARexx-Funktionen bis auf die grundlegenden Kontrollstrukturen und Befehle in einer ARexx-eigenen Funktionsbibliothek zusammengefaßt. Der Interpreter "weiß" also nicht, was eine bestimmte Funktion macht: Er reicht sie nur nach und nach an alle verfügbaren Funktionsbibliotheken und Hosts weiter, bis schließlich eine der Bibliotheken die Funktion für ihn ausführen kann, oder er einen Syntax-Fehler melden muß.

Auf einem Multitasking-System wie dem Amiga kann dieses Konzept besonders vorteilhaft eingesetzt werden: So kann aus einer Applikation heraus ein Makroprogramm gestartet werden, das dann eine andere Applikation "von außen" steuert. Bei der Anwendung sind der Phantasie keine Grenzen gesetzt: So ist es durchaus möglich, beispielsweise aus einer Textverarbeitung heraus auf die Daten einer Datenbank zuzugreifen. Voraussetzung hierfür ist aber, das beide Programme die entsprechende Schnittstelle besitzen.

Der ARexx-Interpreter

Das Kernstück des ARexx-Systems ist der Interpreter, der benötigt wird, um ein ARexx-Programm auszuführen. Im alten AmigaBasic war der zugehörige Interpreter beispielsweise mit dem Editor zu einer funktionalen Einheit verbunden, während das berühmte GfA-Basic auch einen Stand-Alone Interpreter zur Verfügung stellte.

Bei ARexx hingegen ist der Interpreter ein residenter Prozess, der komplett im Systemhintergrund abläuft. Normalerweise wird er durch das Programm 'RexxMast' gestartet. Wenn wir ihm mitteilen wollen, daß wir gerne ein ARexx-Programm ausführen möchten, brauchen wir einen "Vermittler", den Programm-Starter 'RX'.

Von der praktischen Anwendung her gleicht 'RX' dem AmigaDOS-Befehl 'Execute': Als Argument wird der Name einer Stapeldatei angegeben, die dann ausgeführt wird. Im Gegensatz zu den AmigaDOS-Stapeldateien werden die ARexx-Befehle aber direkt vom RexxMaster ausgewertet.

PAG Sandini

Die Struktur des ARexx-Programms

ARexx-Programme sind zunächst einmal Textdateien, die mit einem beliebigen Editor oder auch auf der Kommandozeile geschrieben werden können. Die erzeugten Texte sind direkt lauffähig und müssen nicht erst übersetzt werden. Damit der Interpreter sie von normalen Textdateien unterscheiden kann, müssen sie jedoch unbedingt mit einer Kommentarzeile beginnen. ARexx benutzt zur Eingrenzung von Kommentaren genau wie C die Zeichenfolgen '/*' und '*/', die auch beliebig tief geschachtelt sein dürfen. Der Tradition gemäß enthält der einleitende Kommentar den Programmnamen, oder eine kurze Funktionsbeschreibung.

ARexx sucht ARexx-Programme im logischen Verzeichnis REXX:, wobei es die Dateinamenendung dem Kontext anpaßt. Von der Kommandozeile mit 'RX' zu startende Programme haben die Endung '.rexx', während die Endung für Anwendungs-spezifische Programme von der Anwendung, die sie starten soll, abhängt. So erwartet CygnusEd Dateien mit dem Suffix '.ced', Art Department Professional den Suffix '.adpro', und TurboText die Endung '.ttx'.

Die ersten Schritte

Da wir bekanntlichermäßen durch Praxis besser lernen als durch bloßes Herunterbeten von Regeln und Features, wollen wir im folgenden davon ausgehen, daß Sie ARExx besitzen und erfolgreich installiert haben. Auch die Bedienung Ihres Editors sollte Ihnen keine Probleme bereiten, damit Sie die Beispiele gleich eintippen und ausprobieren können.

Als ersten ARExx-Befehl werden wir 'say', das Äquivalent zum Basic-Befehl "Print" kennenlernen. Wir benutzen 'say', in dem wir den Befehl als erste Anweisung in eine Zeile schreiben, und dann die auszugebende Zeichenkette angeben. Also wollen wir ihn in unserem ersten kleinen ARExx-Programm gleich einmal austesten:

```
/* Hello, World! */  
say "Hello, World"  
exit
```

PAG Sandini

Ein voll kommentiertes und lauffähige ARExx-Programm.

Wenn Sie die paar Zeilen abgetippt haben, dann speichern Sie sie z.B. als "Hallo.Rexx" im Rexx:-Verzeichnis ab. Nun können Sie das Programm per 'RX Hallo' starten. Falls Sie die WShell von Bill Hawes besitzen sollten, genügt auch ein einfaches "Hallo". A propos WShell: An dieser Stelle möchte ich noch einmal auf dieses andere Produkt von William Hawes hinweisen. Die WShell bietet wie am obigen Beispiel ersichtlich einen transparenten ARExx-Support, und viele weitere Extras wie z.B. intelligente Aliase, die selbst wieder ARExx-Kommandos enthalten können. Die WShell ist entweder direkt beim Hersteller erhältlich, oder aber mit einem deutschen Handbuch bei CompuStore.

Doch zurück zu unserem Programm: Es sollte nach dem Start einfach "Hello, World!" in der Shell ausgeben, und dann beenden. Das zusätzliche 'exit', das das Programmende erzeugt, hätte auch weggelassen werden können, da ARExx-Programme automatisch beendet werden, wenn sie die letzte Anweisungszeile bearbeitet haben. Der Klarheit halber haben wir es hier jedoch eingebaut.

Klauseln und Zeilen

Ein Befehl wie 'exit', oder die Kombination aus Befehl und Argument wie 'Say "Hello, World!"' wird in ARexx als Klausel bezeichnet. Neben diesen einfachen Befehlsklauseln kennt ARexx auch Zuweisungsklauseln, in denen einer Variablen ein Wert zugewiesen wird, sowie Kommandoklauseln, die Befehle für externe Programme beinhalten. Da ARexx prinzipiell eine zeilenorientierte Programmiersprache ist, beendet das normale Zeilenende eine Klausel eindeutig. Endet eine Zeile jedoch mit einem Komma, so ist dies für den Interpreter das Zeichen, daß die aktuelle Klausel noch nicht beendet ist, und daß sie in der nächsten Zeile fortgesetzt wird. So können extrem lange Klauseln bequem auf mehrere Zeilen aufgeteilt werden, was die Lesbarkeit eines ARexx-Programms erhöht. Wenn mehrere Klauseln in eine Zeile gepackt werden sollen, so müssen sie durch Semikola getrennt sein. Im Gegensatz zu Pascal ist das Semikolon am Zeilenende jedoch nicht erforderlich.

Literale

PAG Sandini

Ein Literal ist ein Text oder ein Zahlenwert, der direkt als solcher übernommen werden kann. Die Meldung "Hello, World!" in unserem ersten Programm ist beispielsweise ein Zeichenketten-Literal. Zeichenketten werden in ARexx wahlweise mit einem Paar Apostrophen oder Anführungszeichen zusammengefaßt. Obwohl beide Zeichen funktionell identisch sind, dürfen sie nicht gemischt verwenden, um beispielsweise ein Literal mit einem Apostroph einzuleiten, und mit einem Anführungszeichen zu beenden. Dafür darf innerhalb einer Zeichenkette das nicht zur Eingrenzung genutzte Zeichen beliebig oft auftauchen; es grenzt keine Unterzeichenkette ab. Anders jedoch bei dem zur Eingrenzung verwendeten Zeichen: Wenn es in der resultierenden Zeichenkette erscheinen soll, dann muß es doppelt eingegeben werden. Durch Nachstellen eines kleinen x wird eine Zeichenkette als Hex-String gekennzeichnet: Der String enthält dann die hexadezimale Darstellung seines eigentlichen Inhalts.

Konstante Zahlenwerte dürfen sowohl direkt als auch als Text-Literale geschrieben werden. Im letzteren Fall werden bei einer arithmetischen Operation der Zahl voranstehende und nachfolgende Leerzeichen automatisch entfernt.

Verkettung von Zeichenketten

Zeichenketten in Form von Literalen und Variablen können in ARexx mit dem Verknüpfungs-Operator '||' direkt miteinander verkettet werden. Darüber hinaus kennt ARexx jedoch auch die implizite Verkettung, die dann zum Zuge kommt, wenn zwei Zeichenketten direkt aufeinander folgen: Ist mindestens ein Leerzeichen zwischen ihnen, so sind sie in der resultierenden Zeichenkette mit genau einem Leerzeichen getrennt. Direkt benachbarte Zeichenketten werden ohne Zwischenraum aneinander gefügt.

Variablen

Variablen sind im Gegensatz zu Literalen nur Platzhalter für Zahlen und Texte, deren Wert sich während des Programmablaufs ändern kann. Während andere Programmiersprachen für unterschiedliche Daten verschiedene Datentypen bereitstellen, benutzt ARexx die Zeichenkette als uniformen Datentyp für alle Variablen. Die Deklaration von Variablen ist nicht erforderlich, und eine Typtrennung ist unbekannt, so daß die Konvertierung kein Problem darstellt. Der Vorteil liegt auf der Hand: Zahlen und Text können zusammen eingelesen und verarbeitet werden. Die maximale Länge einer Variablen ist auf 65335 Zeichen beschränkt, wobei der String auch Null-Bytes enthalten darf, so daß auch Binärdaten verarbeitet werden können.

Für Variablennamen gelten die üblichen Einschränkungen: Sie müssen mit einem Buchstaben beginnen, und dürfen darüber hinaus Zahlen sowie die Sonderzeichen "?!\$" enthalten. Nationale Sonderzeichen wie z.B. Umlaute sind allerdings nicht zulässig!

Im Gegensatz zu Pascal spielt die Groß/Kleinschreibung von Variablen bei ARexx keine Rolle, da die Variablennamen intern automatisch mit Großbuchstaben dargestellt werden. Nicht initialisierte Variablen sind nicht leer, sondern enthalten immer ihren Namen als Zeichenkette. Mit der Anweisung 'DROP <Variable>' kann eine Variable in ihren uninitialisierten Zustand versetzt werden. Die Zuweisung eines Wertes an eine Variable erfolgt immer über ein Gleichheitszeichen, bei dem links die Variable, und rechts der Wert oder Ausdruck, der zugewiesen werden soll, angegeben ist.

Doch schauen wir uns den Einsatz des oben Gesagten anhand eines kleinen Beispiels an:

```
/* Zeichenketten */
```

```
Beep = '07'x
```

```
say 'Here' "I" 'am!' Beep
```

```
say "Nicht definierte" variable
```

```
variable = 'Variable'
```

```
say 'Definierte' Variable
```

```
exit
```

Mit dem ASCII-Steuercode 7 wird die Beep-Funktion des Systems ausgelöst, die auf einem Standard-Amiga den Bildschirm kurz aufblitzen läßt, wenn kein akustischer Beep definiert wurde.

Mit 'Beep' führen wir in der ersten Zeile eine Variable ein, der wir das Zeichen mit dem ASCII-Code 7 zuweisen. In der zweiten Zeile demonstrieren wir, wie die aufgeführten Literale und die Variable zusammen ausgegeben werden. Die Ausgabe des Zeichens in Beep löst dabei ein Piesen oder ein Bildschirm-Blitzen aus. In der nächsten Zeile sehen wir dann ein Literal sowie eine nicht definierte Variable. Die nächsten zwei Zeilen zeigen dann, wie das Ganze mit einer definierten Variable aussieht.

Eingabe

Natürlich wollen wir mit ARexx nicht nur Daten ausgeben, sondern vor allen Dingen auch erst einmal zur Verarbeitung einlesen. Für die direkte Eingabe von der Kommandozeile bietet sich der Befehl 'pull' an, dem ein Variablenname nachgestellt wird. Bei der Ausführung wartet 'pull' dann auf eine Benutzereingabe, und überträgt den eingelesenen Wert in die angegebene Variable. Beispielsweise so:

```
/* Daten Einlesen */
```

```
say 'Bitte etwas eingeben:'
```

```
pull daten
```

```
say 'Die Eingabe war:' daten
```

```
exit
```

Ein einfaches Programm mit Eingabe und Ausgabe.

Operatoren

In der folgenden Abbildung 4 sind alle mathematischen und textuellen Operatoren von ARexx nebst ihrer Priorität aufgelistet. Besonderes Augenmerk sollte den Vergleichsoperatoren gelten: Bei Relations-Vergleichen wie "Größer", "Kleiner" und "Ist Gleich" werden voranstehende und nachfolgende Leerzeichen nicht beachtet. Die einzige Ausnahme hiervon sind die Vergleiche auf exakte Gleichheit, die auch diese Leerzeichen mit berücksichtigen.

Ausdrücke werden nach Priorität und von links nach rechts ausgewertet. Im Gegensatz zu C werden bei ARexx logische Ausdrücke auch dann noch vollständig ausgewertet, wenn das Ergebnis eigentlich schon feststeht.

Operator	Priorität	Funktion
~	8	Not-Operator
+, -	8	Vorzeichen-Operator
**	7	Potenzierung
*	6	Multiplikation
/	6	Division
%	6	Ganzzahlige Division
//	6	Modulo-Operator
+	5	Addition
-	5	Subtraktion
	4	Verkettung (Auch implizit!)
==	3	Exakte Gleichheit
~=	3	Exakte Ungleichheit
=	3	Gleichheit
~=	3	Ungleichheit
>	3	Größer-Relation
>=, ~<	3	Größer/Gleich-Relation
<	3	Kleiner-Relation
<=, ~>	3	Kleiner/Gleich-Relation
&	2	Konjunktion (AND-Operator)
	1	Disjunktion (OR-Operator)
^, &&	1	Exklusiv-Oder

Liste der Operatoren und ihrer Prioritäten.

Als nächstes wollen wir einige der mathematischen und textuellen Operatoren zusammen mit 'say' erproben. Bitte beachten Sie hier auch die verschiedenen Methoden, mit denen die Zeichenketten zur Ausgabe verknüpft werden.

```
/* Operatoren-Test */
```

```
say 'Bitte geben Sie Ihren Namen ein!'
```

```
pull Name
```

```
say 'Hallo' Name '!'
```

```
say 'Das geht auch, ' || Name
```

```
say '2 + 3 * 5 =' 2+3*5
```

```
say '(2+3) * 5 =' (2+3)*5
```

```
a = 22; b = 7
```

```
say a ' geteilt durch' b '=' a/b
```

```
say '...oder auch' a%b ' Rest' a//b
```

```
say b*b '=' b**2
```

```
c = ' 3.5 '
```

```
say c
```

```
c = +c
```

```
say c
```

```
exit
```

Einige der Operatoren von ARexx in Aktion.

Die Liste läßt sich natürlich noch beliebig fortsetzen, doch dies sei Ihrer eigenen Kreativität überlassen. Schließlich lernt man ja nur durch Praxis. Aus dieser Übung sollten wir vor allen Dingen ein Phänomen im Gedächtnis behalten: Bei der Addition zweier Zahlen spielt es keine Rolle, ob in den Zahlenvariablen Leerzeichen enthalten sind, denn vor der eigentlichen Addition werden sie entfernt, und so enthält das Ergebnis die Leerzeichen auch nicht mehr.

Diesen Effekt können wir nutzen, wenn wir die Leerzeichen aus einer Zahlenvariablen entfernen wollen: Wir weisen einfach der Variablen ihren eigenen Wert zu, und stellen dabei ein Plus als Vorzeichen voran.

Befehlsblöcke

Nachdem wir uns zunächst einmal grundlegend mit den Variablen und Operatoren vertraut gemacht haben, wollen wir uns weiteren Befehlen zuwenden: Zuerst werden wir einige grundlegende Strukturbefehle kennenlernen. Eine wesentliche Eigenschaft von Arexx ist die Möglichkeit, Befehlsfolgen in einem Befehlsblock zusammenzufassen. Ein Block wird durch das Schlüsselwort 'Do' eingeleitet, und durch 'End' abgeschlossen. Die Wirkung entspricht der der geschweiften Klammer von C oder der BEGIN..END-Konstruktion von Modula. Zur bedingten Abarbeitung von Programmteilen bietet ARexx die übliche If..Then..Else-Struktur an.

Ein einfaches Beispiel soll die Verwendungsweise illustrieren:

```
/* If-Demo */
```

```
say 'Bitte A eingeben: '; Pull a
say 'Bitte B eingeben: '; Pull b
```

```
if a > b Then
    say 'A ist größer als B'
else
    say 'B ist größer als A'
```

```
if b > a then do
    say 'A =' a
    say 'B =' b
    say 'A ist größer als B'
end
else do
    say 'A =' a
    say 'B =' b
    say 'A ist größer als B'
end
```

```
exit
```

Die bedingte Abarbeitung von Programmteilen, demonstriert mit dem If-Konstrukt von ARexx. Wenn mehrere Anweisungen bedingt verarbeitet werden sollen, müssen sie in einem Anweisungsblock zusammengefasst werden.

'if' erhält als Argument einen logischen Ausdruck. Wenn dieser Ausdruck logisch wahr war, dann wird die auf das Schlüsselwort 'then' folgende Anweisung ausgeführt. Ansonsten wird eben diese Anweisung übersprungen. Wenn das Schlüsselwort 'else' vorhanden ist, wird die nachfolgende Anweisung ausgeführt, wenn die bei 'If' angeführte Bedingung falsch war, während sie ansonsten ebenfalls übersprungen wird.

Bei geschachtelten If-Anweisungen kann 'Else' allerdings zu Problemen führen: Was macht man, wenn eine der inneren If-Anweisungen kein Else-Statement benötigt, das nächste äußere If-Statement aber schon? Hierzu gibt es eine besondere Anweisung, die überhaupt nichts tut: 'Nop'. Schauen wir uns den Einsatz an einem konstruierten Beispiel an:

/ Nop-Demo */*

say 'Bitte A eingeben: '; Pull a

say 'Bitte B eingeben: '; Pull b

say 'Bitte C eingeben: '; Pull c

if a <= b then

if b <= c then

say 'a <= b <= c'

else

nop

else

if b > c then

Say 'a > b > c'

exit

Richtiges Zusammenfassen von If- und Else-Anweisungen mit der Instruktion nop.

Wenn bei der inneren If-Abfrage kein 'Else Nop' vorhanden wäre, dann würde das nachfolgende 'Else' als alternative Bedingung der inneren If-Abfrage angesehen, was natürlich zu einem nicht beabsichtigten Effekt führen könnte.

Schleifenkonstrukte

Die Anweisung 'do' ist jedoch weit mächtiger, als bisher angedeutet: In ihr sind alle gängigen Schleifenvarianten integriert. Als einfachstes Beispiel wollen wir uns einmal das ARexx-Äquivalent der üblichen For-Schleife anschauen:

```
/* For-Schleife */
```

```
say 'Wie oft soll ich 'Hallo!' ausgeben?'
```

```
pull Zahl
```

```
do i=1 to Zahl
```

```
say 'Hallo!'
```

```
end
```

```
say 'Und jetzt noch einmal halb soviel...'
```

```
do i=1 To Zahl By 2
```

```
say 'Hallo!'
```

```
end
```

```
exit
```

PAG Sandini

Mit der do-Schleife können Anweisungsblöcke wiederholt ausgeführt werden.

Mit dem ersten Argument sagen wir 'do', welche Variable wir als Laufvariable benutzen möchten. Durch die Zuweisung initialisieren wir die Variable mit einem Startwert. Danach wird der Schleifenkörper solange durchlaufen, bis der Grenzwert erreicht ist. Wie sie sicherlich schon erraten haben, folgt der Grenzwert der Schleife nach dem Schlüsselwort 'to'. Zusätzlich können wir nach dem optionalen Schlüsselwort 'by' eine Schrittweite vorgeben, die nach jedem Durchlauf zur Laufvariablen addiert wird. Die Schrittweite kann durchaus auch negativ sein, wobei dann zu beachten ist, daß der Zielwert niedriger als der Startwert sein sollte. Wenn wir auf das Schlüsselwort 'by' verzichten, so beträgt die Schrittweite automatisch 1. Oft gibt es auch den Fall, daß wir überhaupt keine Laufvariable benötigen; wie z.B. im obigen Testprogramm. Dann bietet 'do' in Kombination mit dem Schlüsselwort 'for' sich als Alternative an:

/* Einfacheres For */

say 'Wie oft soll ich 'Hallo!' ausgeben?'

pull Zahl

do for Zahl

say 'Hallo!'

end

exit

Die einfachere Do-For Schleife.

Das Argument nach 'for' legt fest, wie oft die Schleife durchlaufen werden soll. Allerdings sind nur ganze Zahlen als Argumente zugelassen! In anderen Fällen soll die Schleife eine unbestimmte Zeit lang durchlaufen werden, oder gar für immer. Dafür gibt es eine iterative Version des Befehls 'do', die keine Art von Schleifenzähler kennt. Sie wird durch das Schlüsselwort 'forever' deklariert.

PAG Sandini

/* Endlosschleife */

say 'Bitte "Nein" eingeben, um abzubrechen!'

do forever

say 'Hallo'

say 'Nochmal?'

pull Antwort

if Antwort = Nein then leave

end

exit

Eine Endlosschleife mit Abbruchbedingung.

Der aufmerksame Leser wird den neuen Befehl 'leave' sofort erkannt haben, und auch seine Funktionsweise erraten haben: Er unterbricht die Abarbeitung der aktiven Schleife, und setzt die Programmausführung nach dem abschließenden End fort. In geschachtelten Schleifen kann er sogar mehrere Ebenen nach oben springen, wenn als Argument eine Laufvariable angegeben wird. Dann nämlich verläßt er die Schleife, die diesen Schleifenzähler benutzt.

Beachtung verdient auch der Operator 'Nein' des If-Statements. Wir benutzen ihn hier nicht als Zeichenkette, sondern als nicht initialisierte Variable, die deshalb den Wert 'NEIN' hat. Beim Einlesen von Variablen mit 'Pull' werden Texte automatisch in Großschrift gewandelt. Wer dies jetzt für einen großen Nachteil von Arexx hält, dem sei versichert, daß zum Einlesen von Texten eigentlich ein anderer Befehl benutzt werden sollte, den wir in dieser Folge aber wohl nicht mehr kennenlernen werden.

Nicht immer ist es elegant, eine Schleife mit 'leave' zu verlassen: ARexx bietet neben 'For' und 'Forever' noch zwei weitere iterative Schleifentypen: Die eine wird durchlaufen, bis eine vorgegebene Bedingung erfüllt ist, während die andere nur durchlaufen wird, solange die Bedingung gültig ist. Schauen wir uns zunächst die erste Variante an. Da sie den 'Repeat..Until'-Schleifen anderer Programmiersprachen entspricht, wird die Abbruchbedingung nach dem Schlüsselwort 'Until' angegeben.

/* Repeat..Until Emulation */

say 'Bitte A eingeben: '

pull a

say 'Bitte B eingeben: '

pull b

do until a = b

if a < b then do

c = a

a = b

b = c

end

a = a - b

end

say 'Der ggT ist' a

exit

Das Programm zur Berechnung des größten gemeinsamen Teilers zweier Zahlen demonstriert eine weitere Schleifenvariante.

Der ggT ist natürlich der größte gemeinsame Teiler, wie Sie sich sicherlich er-

PAG Sandini

innern werden. Der verwendete Algorithmus ist denkbar einfach: Wenn ich den größten gemeinsamen Teiler zweier Zahlen ermitteln will, ziehe ich solange die kleinere Zahl von der größeren ab, bis beide gleich sind. Das Endergebnis ist der größte gemeinsame Teiler der beiden Zahlen. Ist das Ergebnis 1, so sind die Zahlen teilerfremd, d.h. es gibt keine andere Zahl, die die beiden angegebenen Zahlen teilt.

Die Programmbeschreibung fordert von uns, daß wir die größere Zahl von der kleineren abziehen, also müssen gegebenenfalls beide Zahlen vertauscht werden. Dies erledigt die 'If'-Anweisung und die drei nachfolgenden Wertzuweisungen.

Der letzte Schleifentyp entspricht der bekannten und geliebten 'While'-Schleife. Die Konstruktion ist mit der 'Until'-Schleife identisch, bis auf die Abbruchbedingung, was wir uns in einem Beispiel anschauen:

/ Schleifenvergleich */*

x = 0; z = 1

do until x ~= z

say 'Until durchlaufen!'

end

do while x = z

say 'While durchlaufen!'

end

exit

Die Until- und While-Schleifen im Vergleich.

Das Ergebnis wird manchen vielleicht (nicht) erstaunen: Die Schleifenbedingung war bei beiden Schleifen falsch - und trotzdem wurde der Anweisungsblock der 'Until'-Schleife durchlaufen, während er bei der 'While'-Variante nicht beachtet wurde. Dies liegt an der Art der Auswertung: Bei 'Until' wird die Durchlaufsbedingung am Ende der Schleife abgefragt, während bei 'While' dieser Test im Schleifenkopf stattfindet.

Nachdem wir den 'Leave'-Befehl zum Verlassen von Programmschleifen kennengelernt haben, sei hier noch ein kleiner Nachtrag erlaubt: natürlich

gibt es auch ein Äquivalent für nicht iterative Anweisungsblöcke; den Befehl 'Break'. Er findet übrigens nicht nur hier Anwendung, sondern wir werden ihn auch später noch einmal wiedersehen.

Die Stem-Variablen

Nachdem wir nun einige der grundlegenden Programmkonstrukte von ARexx kennengelernt haben, wollen wir uns nun den speziellen Feinheiten zuwenden: Eine besondere Eigenheit von ARexx sind die Stem-Variablen, deren Funktion denen eines Arrays in anderen Programmiersprachen nahekommt. Die Stem-Variable besteht aus einem Stammnamen, dem eine oder mehrere durch Punkte voneinander abgetrennte Komponenten-Variablen folgen können. Bei der Interpretation werden die Inhalte der Komponenten-Variablen ausgewertet, und so ein neuer Variablenname gebildet, der dann adressiert wird.

Beschränkt man sich bei der Belegung der Komponenten-Variablen auf Zahlen, so arbeitet man mit einem gewöhnlichen Array, doch die Stärke von ARexx liegt in der Tatsache, daß sich so auch assoziative Speicher realisieren lassen. Wenn wir beispielsweise eine Liste von Namen und Telefonnummern ablegen wollten, dann müßten wir in einer der klassischen Programmiersprachen zwei Felder anlegen, in denen Name und Nummer unter jeweils dem selben Index gespeichert sind. Wenn wir auf eine bestimmte Nummer zugreifen wollten, so müßten wir zunächst das Namensfeld durchsuchen, bis der passende Eintrag gefunden ist, und dann aus dem Nummern-Feld das korrespondierende Element auslesen. Unter ARexx bedienen wir uns einfach einer Stem-Variablen, in der der Name als Index auf die Nummer dient. Einfacher geht es nicht mehr!

Darüber hinaus haben Stem-Variablen noch zwei weitere angenehme Eigenschaften: Durch Benutzung des Stammsnamens können alle zugehörigen Variablen initialisiert oder ge-'DROP'-t werden.

/* Telefon-Buch */

Nr. = '???'

Nr.Thomas = '29872'

Nr.Verlag = '82299'

```
say "Name ?"  
pull Name /* einlesen */  
  
if Nr.Name = '???' then  
say 'Nicht gefunden!'  
else  
say 'Die Nummer von' Name 'ist' Nr.Name
```

Ein einfaches elektronisches Telephonbuch, das Stem-Variablen als assoziativen Speicher benutzt.

If und seine Alternativen

Bei komplexen Fallunterscheidungen wird es oft recht mühsam, eine passende Konstruktion aus geschachtelten If-Anweisungen zu finden, die auch wirklich alles berücksichtigt. Mit den 'select'-Statement bietet ARexx eine sehr komfortable Alternative zu den sonst üblichen 'Case'-Anweisungen für mehrfache Auswahlen: 'select' leitet einen Auswahlblock ein, der mit 'end' abgeschlossen werden muß.

Innerhalb dieses Blocks werden die einzelnen Auswahlen in der Form 'when <Ausdruck> then <Anweisung>' aufgezählt. Als letzte Anweisung im Block sollte mit 'otherwise' eine Standard-Auswahl angegeben werden. Sie ist zwar nicht zwingend erforderlich, aber andererseits wird ein Fehler gemeldet, wenn sie fehlt, und keine der vorangegangenen 'when'-Anweisungen zuge troffen ist. Im Zweifelsfall schadet ein kurzes 'otherwise nop' nicht.

Im Gegensatz zu den üblichen 'CASE'-Anweisungen müssen sich die 'when'-Statements nicht alle auf eine bestimmte Variable beziehen: Es ist wirklich jede Art von Bedingung bei der Abfrage zulässig.

Ein- und Ausgabe

In allen REXX-Implementationen stehen zumindest die Befehle 'say' und 'pull' zur Ein- bzw. Ausgabe zur Verfügung. Sie bedienen sich dabei der Standard Ein- und Ausgabekanäle "stdin" und "stdout", die ein Shell-Fenster

zur Verfügung stellt. Normalerweise kann von der Standard-Eingabe nur gelesen werden: Im Fall einer Shell kann ein ARexx-Programm die Standard-Eingabe mit den Kommandos 'queue' und 'push' auch beschreiben. 'queue' und 'push' unterscheiden sich dabei in der Art und Weise, wie die Daten in den Strom geschrieben werden:

Bei 'push' kann die zuerst geschriebene Zeile auch wieder zuerst ausgelesen werden, während bei 'queue' die zuletzt geschriebene Zeile an erster Stelle im Ausgabestrom steht. Bei einem aus der Shell gestarteten ARexx-Programm landen Zeilen, die in der Schlange stehengeblieben sind, weil sie zwar geschrieben, aber nicht mehr gelesen wurden, als voraus getippte Befehle in der Kommandozeile!

```
/* Remote Dir */  
push 'dir sys:'  
exit
```

Mit 'push' und 'queue' läßt sich der Standard-Eingabestrom beschreiben.

PAG Sandini

Optionen

Mit dem 'options'-Befehl von ARexx können gewisse Voreinstellungen festgelegt werden: Nach 'options failat' ist der Grenzwert anzugeben, ab dem ein Return-Code einen Fehler auslöst, der zum Programmabbruch führt. Von der Funktion her entspricht diese Option also dem 'Failat'-Kommando der AmigaDOS-Batchfiles.

Nach 'options prompt' kann ein Prompt-String vorgegeben werden, der dann als Eingabeaufforderung bei jeder Benutzung von 'pull' erscheint.

Schließlich bleibt uns noch der Befehl 'options results', mit dem man ARexx auffordert, Rückmeldungen von externen Befehlen abzufragen und in der Variablen 'result' zu speichern.

Ablaufverfolgung

Beim Entwickeln von ARexx-Programmen wird zumindest Anfangs viel Zeit für die Fehlersuche verwandt. Eine große Hilfe dabei sind die Trace-

Funktionen von ARexx, die durch den Befehl 'trace' mit verschiedenen Optionen aktiviert werden. Im Rahmen dieses Schnelleinstiegs wollen wir uns nur einige der möglichen Anwendungen anschauen; für eine ausführliche Beschreibung aller Fähigkeiten des 'trace'-Befehls muß auf das "ARexx User's Reference Manual", Kapitel 7, verwiesen werden.

Damit der Tracing-Output nicht mit irgendwelchen Ausgaben des zu überprüfenden Programms gemischt wird, öffnen wir zunächst das ARexx-Tracing-Fenster mit dem CLI-Kommando 'TCO' (Tracing Console Open). Nach Gebrauch können wir es später mit 'TCC' (Tracing Console Close) wieder schließen. In dem zu überprüfenden Programm muß der Trace-Modus aktiviert werden: Für den Anfang sollte die Zeile 'trace results' gleich nach dem einleitenden Kommentar ausreichend sein. Obengenannte Option sorgt dafür, daß alle Zeilen in der Reihenfolge der Ausführung im Tracing-Fenster landen, und daß zudem die Ergebnisse aller Zuweisungen und Berechnungen angezeigt werden.

Wer auch Informationen über alle Zwischenergebnisse haben möchte, die bei der Auflösung eines Ausdrucks auftauchen, der sollte statt der Option 'results' die Option 'intermediates' angeben.

Indem man den oben genannten Optionen ein Fragezeichen voranstellt, aktiviert man den interaktiven Tracing-Modus: Dann wartet der Interpreter nach jeder Programmzeile auf eine Eingabe des Anwenders. Eine einfache Bestätigung mit Return setzt das tracing in der nächsten Zeile fort, während die Eingabe eines Gleichheitszeichens mit anschließendem Return die letzte Zeile noch einmal ausführt. Alle anderen Eingaben werden als ARexx-Befehl interpretiert, der jetzt sofort ausgeführt werden soll. So könnte z.B. der Wert einer Variablen neu zugewiesen werden. Zu beachten ist allerdings, daß Schleifen oder ähnliche Konstrukte in der Eingabezeile beendet werden müssen.

Unterprogramme

Die Möglichkeit, Unterprogramme zu definieren, bildet ein wichtiges Merkmal einer strukturierten Programmiersprache. Bei ARexx geht es besonders einfach, denn prinzipiell genügt es, den Einsprungspunkt eines Unterprogramms durch eine Sprungmarke zu definieren. Die Sprungmarke, auch Label genannt, muß nach den gleichen Regeln wie ein Variablenname

gebildet werden; zusätzlich wird ihr noch ein Doppelpunkt nachgestellt. Der Aufruf erfolgt prinzipiell einmal mit dem Statement 'call <Label>', womit zur Sprungmarke verzweigt wird. Wird nach dem Label das Schlüsselwort 'procedure' angegeben, so verwendet die Prozedur lokale Variablen.

Die Variablen des Hauptprogramms sind somit für die aufgerufene Routine unsichtbar; mit dem Schlüsselwort 'expose <varname>' können jedoch einzelne Variablen des aufrufenden Programms für das Unterprogramm sichtbar gemacht werden. Vorsicht ist bei Stem-Variablen geboten:

Zwar kann durch Angabe des Stamm-Namens der ganze Stem sichtbar gemacht werden; problematisch wird es jedoch, wenn nur Teile des Stems sichtbar sein sollen.

Dann kommt es unter Umständen auf die Reihenfolge an, mit der die Variablen sichtbar gemacht werden.

/* Sichtbarkeit */

B = 'Symbol'

A.B = 'Stem-Symbol'

PAG Sandini

call Test1

call Test2

exit

Test1: PROCEDURE EXPOSE A.B B

say 'A.B wird vor B sichtbar:'

say 'A.B =' A.B 'B =' B

return

Test2: PROCEDURE EXPOSE B A.B

say 'B wird vor A.B sichtbar:'

say 'A.B =' A.B 'B =' B

return

In der Prozedur 'Test1' wird A.B vor B sichtbar. Da B zu diesem Zeitpunkt nicht initialisiert ist, kann auch nicht auf den Inhalt aus A.B aus dem Hauptprogramm zugegriffen werden, da dort A.B nur für ein B mit dem Inhalt 'Symbol' definiert ist.

Wie am Beispiel schon ersichtlich, muß eine Prozedur mit 'return' abgeschlossen werden. Nach diesem Schlüsselwort kann man auch einen Rückgabewert angeben, der bei einem Aufruf mit 'call' der Systemvariablen 'result' zugewiesen wird. Alternativ kann man selbstdefinierte Prozeduren auch als Funktion aufrufen, wobei dann dem Prozedurnamen ein Klammernpaar nachgestellt werden muß. Der Rückgabewert kann dann einer beliebigen Variable zugewiesen werden.

Natürlich können einer Funktion auch Argumente übergeben werden: Die einfachste Methode sie auszuwerten ist die Verwendung der Funktion 'ARG(<n>)', die das n-te Argument an die Funktion zurückliefert. ARexx-Programme selbst sind in diesem Zusammenhang auch Prozeduren, die so auf ihre Argumente der Kommandozeile zurückgreifen können.

Bei der Verwendung von selbstdefinierten Prozeduren ist es guter Programmierstil, das Hauptprogramm vor den Prozeduren mit 'exit' zu verlassen, da der Interpreter bei der Abarbeitung des Programms sonst in den Bereich der Unterprogramme hineinläuft.

/* Fakultät */

options prompt 'Bitte ganze positive Zahl eingeben: '

do until Zahl > 0 & Zahl = Zahl % 1

PULL Zahl

end

say 'Die Fakultät von' Zahl 'ist' Fak(Zahl)

exit

Fak: procedure

x = arg(1)

if x > 1 then x = x * Fak(x-1)

return x

Das rekursive Programm zur Berechnung von Fakultäten nutzt die Übergabe von Argumenten und Resultaten.

Parsing

Eine andere Art der Auswertung von Argumenten bietet die sehr mächtige 'parse'-Anweisung von ARExx. Da eine vollständige Besprechung aller ihrer Möglichkeiten den Rahmen dieses Schnelleinstiegs sprengen würde, werden wir nur einige Einsatzmöglichkeiten erwähnen. Eine vollständige Beschreibung von 'parse' findet sich im "ARExx User's Reference Manual" in Kapitel 8.

Mit 'parse arg <variable 1> ... <variable n> .' können die übergebenen Argumente wortweise den n aufgeführten Variablen zugewiesen werden. Liefert das Argument mehr Worte, als Variablen angegeben wurden, so sorgt der nachgestellte Punkt dafür, daß der Rest der Argumente ignoriert wird. Wird er weggelassen, so weist 'parse' der letzten Variablen den gesamten Rest des Argumentstrings zu. Es gilt zu beachten, daß 'parse' nicht mit String-Tokens zurecht kommt: Soll "Dies ist ein Test" geparkt werden, so liefert 'parse' die Tokens "'Dies', 'ist', 'ein' und 'Test'", anstatt, wie erwartet "'Dies ist ein Test'".

'parse pull' liest für jede der angegebenen Variablen einen String von der Konsole ein. Wird 'pull' - oder irgend einer der anderenen Quelloptionen - das Schlüsselwort 'upper' vorangestellt, so werden die gelesenen Daten automatisch in Großbuchstaben umgesetzt.

'parse var <variable>' benutzt den Inhalt der angegebenen Variablen als Quelle für alle Zuweisungen.

Mit 'parse source' können Informationen über den Aufrufer und den Namen des aktuellen Programms abgefragt werden: Der zum Parsing zur Verfügung gestellte String liefert als erstes Token entweder den String 'COMMAND' oder 'FUNCTION'; in Abhängigkeit davon, wie das aktuelle Programm aufgerufen wurde. An zweiter Stelle folgt mit '0' oder '1' ein Indikator, ob der Aufrufer ein Resultat erwartet. Danach liefert 'parse source' den Namen, mit dem das Programm ursprünglich gestartet wurde, den vollständigen Dateinamen mit Pfadangabe, den aktuellen Dateisuffix, sowie den Namen der Befehlsumgebung, aus der das Programm heraus aufgerufen wurde. Weitere Informationen über das aktuelle System können mit 'parse version' abgefragt werden: Der Zielstring enthält dann Tokens für Versionsnummer, Bezeichnung für CPU und FPU (falls vorhanden), sowie Video-Modus und Videofrequenz.

Interpretation

Als Interpreter-Sprache verfügt ARexx auch über einige ungewöhnliche Features: So kann beispielsweise mit dem Befehl 'interpret <variable>' der Inhalt einer Variablen als ARexx-Befehl ausgeführt werden.

```
/* Interpreter */
```

options prompt "Bitte einen Befehl eingeben: "

```
do until cmd = ""
```

```
pull cmd
```

```
interpret cmd
```

```
end
```

```
exit
```

Die kurze Interpreter-Schleife kann genutzt werden, um interaktiv Befehle auszuprobieren.

String-Manipulation

Nun kann man mit ARexx glücklicherweise mehr machen, als einfach nur zwei Strings miteinander zu verknüpfen: Die grundlegende String-Manipulation ist auch hier das Abschneiden von rechten und linken Teilketten mit den Funktionen 'Left()' und 'Right()'. Das Resultat dieser Funktionen ist immer die gekürzte Zeichenkette, während die Parameter zum einen aus der Original-Zeichenkette bestehen, und zum anderen aus der Anzahl der abzuschneidenden Zeichen. Optional kann als dritter Parameter ein Füllzeichen angegeben werden, mit dem der String am gegenüber liegenden Rand aufgefüllt wird, wenn mehr als seine Länge abgeschnitten wird.

Strings beliebiger Länge können mit 'Copies()' erzeugt werden: Als erstes Argument übergibt man das Zeichen, mit dem der String gefüllt sein soll, und als zweites Argument die Länge des Ergebnis-Strings. So lassen sich auch lange Strings kompakt darstellen! Ein ähnliches Anwendungsgebiet gilt für 'Xrange()', das zwei Zeichen-Argumente erwartet: Der Ergebnis-String ent-

hält dann alle Zeichen, die zwischen den beiden angegebenen Zeichen liegen. Beide Argumente sind optional: Der Start-Wert ist auf das Zeichen '00'x, und der Endwert auf das Zeichen 'ff'x festgelegt.

/* Right() Left() Demo */

**/*
* Baut eine Tabelle mit Resultaten der Funktionen Left()
* und Right(), die deren Funktionsweise demonstriert.
*/**

**say 'Bitte gib deinen Namen ein:'
Name = ReadLn(stdin)
if Length(Name) < 6 then do
say 'Dein Name mußte verlängert werden!'
Name = Name||Left('+++++',6-Laenge)
end**

Laenge = Length(Name)

**Blank= Copies(' ',Laenge)
Line=Copies('-',Laenge)**

**say ' | 'Right' Left(Blank,Laenge-6) '|' 'Left'
say ' —+—' ||Line||'+—' ||Line
do i = 1 to Laenge
say Right(' ',i,2) '|' Left(Name,i) Left(Blank,Laenge-i),
||'|'Right(Blank,Laenge-i)
Right(Name,i) end**

Dieses kleine Beispielprogramm demonstriert auf anschauliche Weise die Funktionen Left() und Right(). Das hier benutzte Copies() erzeugt aus dem angegebenen Zeichen durch Wiederholung eine Zeichenkette der angegebenen Laenge.

Die Funktion SubStr() stellt eine Kombination aus Left() und Right() dar: mit ihr wird aus einem String-Argument <str> ab der Position <start> eine Unterzeichenkette <substr> der spezifizierten <laenge> ausgeschnitten, die

notfalls mit dem Füllzeichen <pad> aufgefüllt wird. Die Syntax:

<substr> = SubStr(<str>,<start>,<laenge> [,<pad>])

Wird das Füllzeichen nicht explizit angegeben, so wird der angeforderte String mit Leerzeichen aufgefüllt. Auch hierzu haben wir wieder ein kleines Beispielprogramm vorbereitet:

/* SubStr() Demo */

Call WriteCh(stdout,'Bitte gib Deinen Namen ein: ')
test = ReadLn(stdin)

say

say ' | 1 | 2 | 3 '

say '—+—+—+—+—'

do i = 1 To Length(Test)

call WriteCh(stdout,Right('i,2)'|')

call WriteCh(stdout,SubStr(test,i,1,'.')'|')

call WriteCh(stdout,SubStr(test,i,2,'.')'|')

call WriteLn(stdout,SubStr(test,i,3,'.'))

end

Ein Beispielprogramm für SubStr().

Ein/Ausgabe

Neben 'push', 'pull' und 'say' kennt AReXX auch Amiga-spezifische Funktionen zur Ein- und Ausgabe von Zeichen und Zeichenketten in Dateien. Die Ein/Ausgabe ins CLI-Fenster erfolgt über die logischen Dateien 'stdin', 'stdout' und 'stderr', die automatisch geöffnet werden. Alle anderen Dateien müssen zunächst mit 'Open()' geöffnet werden. Die Syntax von 'Open()' lautet:

<success> = Open(<kanal>,<name>,<modus>)

<success> ist dabei ein logischer Erfolgswert, der anzeigt, daß die Datei erfolgreich geöffnet wurde. <kanal> ist eine benutzerdefinierte Bezeichnung,

die als Filehandle bei den anderen Dateioperationen genutzt wird. Die Filehandles für die Standard Ein/Ausgabefunktionen gelten als vordefiniert; die entsprechenden Files müssen also nicht explizit geöffnet werden. <name> ist eine Variable oder Zeichenkette, die einen gültigen DOS-Dateinamen enthält. <modus> ist eine Zeichenkette oder Variable, die die Zugriffsart durch eine beliebige Abkürzung für "read", "write" oder "append" kennzeichnet.

Nach Gebrauch sollte man der Ordnung halber jede geöffnete Datei mit 'Close()' schließen, auch wenn der Interpreter dies nach Programmende 'automatisch' erledigt.

Konnte die Datei geöffnet werden, so kann sie je nach Modus mit 'ReadCh()' und 'ReadLn()' oder 'WriteCh()' und 'WriteLn()' gelesen bzw. beschrieben werden. Die Syntax in Kürze:

<string> = ReadCh(<kanal>,<bytes>)

Versucht <bytes> Zeichen aus der Datei mit dem Filehandle <kanal> zu lesen. Wurde die Datei dabei erschöpft, so enthält der String nur so viele Zeichen, wie tatsächlich gelesen wurden.

<string> = ReadLn(<kanal>)

Liest aus der Datei alle Zeichen bis zum nächsten Zeilenvorschub, und legt das Resultat ohne Zeilenvorschub im Zielstring ab.

<bytes> = WriteCh(<kanal>,<string>)

<bytes> = WriteLn(<kanal>,<string>)

Beide Funktionen schreiben die angegebene Zeichenkette in die über das Filehandle angegebene Datei, und liefern die Zahl der geschriebenen Zeichen zurück. Im Gegensatz zu 'WriteCh()' fügt 'WriteLn()' aber einen Zeilenvorschub an.

Die Schreib/Leseposition innerhalb einer Datei kann mit der Funktion 'Seek()' gesetzt und bestimmt werden:

<position> = Seek(<kanal>,<offset>,<anker>)

Seek() verschiebt die aktuelle schreib/Leseposition um <offset> Zeichen relativ zum angegebenen Ankerpunkt: Dies kann mit 'Begin' der Anfang und

mit 'End' das Ende der Datei sein, wenn es nicht mit 'Current' die aktuelle Position ist. Als Ergebnis liefert 'Seek()' immer die tatsächliche Position relativ zum Anfang der Datei zurück.

Das Dateieneinde kann ähnlich wie in C mit der Funktion '<success> = Eof(<kana-
nal>)' abgeprüft werden, die bei Erreichen des Dateieendes eine 1 als Wahrheitswert True zurückliefert.

```
/* Drucke Version String
```

```
*
```

```
* Aufruf: ver <filename>
```

```
*
```

```
*/
```

```
parse arg file .
```

```
if ~Open('in',file,'r') then do
```

```
say 'Kann Datei' file 'nicht öffnen!'
```

```
exit(5)
```

```
end
```

```
/* Dateilänge feststellen */
```

```
filelen = Seek('in',0,'E')
```

```
/* File Blockweise einlesen */
```

```
do filepos = 0 to filelen by 1000
```

```
call Seek('in',filepos,'b')
```

```
data = ReadCh('in',1005)
```

```
verstart = Pos('$VER',data)
```

```
if verstart ~= 0 then do
```

```
/* Auf Anfang von Version-String */
```

```
call Seek('in',filepos+verstart,'b')
```

```
data = ReadCh('in',256)
```

```
/* Ende des Strings finden */
```

```
verend = Pos('00'x,data)
```

```
say Left(data,verend)
```

```
leave filepos
```

```
end
```

```
end
```

```
call Close('in')
```

```
exit
```

PAG Sandini

ver.rexx' demonstriert einige Befehle der Dateiverarbeitung in einem kleinen Utility, das eine Datei nach einem Versionsstring durchsucht, der mit dem Kürzel '\$Ver' eingeleitet wird.

Wort-Operationen

Neben den Operationen auf Zeichenbasis kenn ARexx auch Operationen auf Wort-Basis: Dabei ist die kleinste Verarbeitungseinheit eine Teilzeichenkette, die keine Leerzeichen enthält. So liefert beispielsweise die Funktion 'Words()' die Anzahl der Worte in der Übergebenen Zeichenkette. Zu beachten ist, daß durch Anführungszeichen keine Worte geschaffen werden können, die Leerzeichen enthalten, wie das z.B. auf der Kommandozeile bei CLI-Befehlen möglich ist.

/* Words()-Test */

say Words('Nur Worte zählen!')
say Words('Sehr "v i e l e" Worte')

Durch Anführungszeichen kann keine Teilzeichenkette zum Wort gemacht werden!

Als weitere Operation kann mit 'Word(<string>,<n>)' das n-te Zeichen aus einer Zeichenkette ausgelesen werden. Zwei weitere Funktionen haben die gleiche Syntax zum Aufruf: 'Wordindex()' liefert dann die Position des ersten Zeichens des bezeichneten Words, während 'Wordlength()' die Länge des avisierten Words liefert. Ein Rückgabewert von 0 bedeutet, daß die Zeichen-kette weniger Worte als angefordert besitzt.

Komplexer wird es dann mit 'Subword()', das ein Äquivalent zu 'substr()' darstellt: Die Syntax lautet 'Subword(<string>,<n>)', wobei als optionaler dritter Parameter auch die Anzahl der Worte übergeben werden kann, die aus dem angegebenen String geliefert werden sollen. Wird keine Länge vorgegeben, so liefert die Funktion den Rest der Zeichenkette.

Eine verwandete Funktion ist 'Delword()', das ebenso aufgerufen wird: Diese Funktion liefert den Teil des Strings, den 'Subword()' nicht liefern würde.

Werden 'Delword()' und 'Subword()' mit gleichen Parametern auf den selben String losgelassen, dann kann aus ihren Resultaten der ursprüngliche String wieder zusammengesetzt werden.

Weitere String-Operationen

In Beispiel 11 haben wir eine Funktion benutzt, die in gewisser Weise mit den Wort-Operationen verwandt ist: 'Pos()' liefert die Position einer Zeichenkette innerhalb einer anderen Zeichenkette. Das zu suchende Muster ist dabei erstes Argument, und der zu durchsuchende String das zweite. Als optionaler Parameter kann auch eine Startposition übergeben werden.

Liefert die Funktion 0 als Resultat, so wurde die Zeichenkette nicht gefunden. Wem die Syntax von 'Pos()' nicht gefällt, der kann sich an Index halten; einer funktional gleichwertigen Funktion, bei der nur die ersten beiden Parameter vertauscht sind. Schließlich gibt es mit 'Lastpos()' eine dritte Suchfunktion, die syntaktisch mit 'Pos()' verwandt ist, aber den String von hinten statt von vorne durchsucht.

In gewisser Weise stellt 'Overlay()' die Umkehrung von 'Pos()' dar: das erste Argument ist ein String, der über vorhandene Zeichen im Zeiten String gelegt wird. Im Normalfall wird von vorne her gefüllt, doch als optionaler dritter Parameter kann eine Startposition für den Füllvorgang festgelegt werden. Der optionale vierte Parameter gibt dann an, wieviele Zeichen kopiert werden sollen, während der optionale fünfte Parameter ein Füllzeichen ist, mit dem nicht vorhandene String-Teile gefüllt werden.

/* Overlay()-Demo */

```
do i = 1 to 10  
say Overlay('Special', '**Amiga', i, 8, '**')  
end  
  
exit
```

Mit dem Füll-Zeichen wird gegen Ende der Schleife zwischen den beiden Zeichenketten und am Ende des ersten Strings aufgefüllt.

Auch 'Verify()' ist in gewisser Weise mit 'Pos()' verwandt: Als erstes Argument gibt es eine zu durchsuchende Zeichenkette, doch als zweites Argument nicht etwa einen Suchstring, sondern eine Zeichensammlung. Die Funktion liefert dann die Position des ersten Zeichens im Suchstring, das nicht in der Liste enthalten war. Wird das optionale Schlüsselwort 'match' als dritter Parameter übergeben, so wird die Logik umgekehrt, und die Funktion liefert die Position des ersten Zeichens, das in der Liste enthalten war. Eine Rückgabewert 0 zeigt an, daß die Bedingung nicht zu erfüllen war: Es waren entweder alle oder kein Zeichen aus der Liste im Suchstring.

Mit 'Translate()' können Zeichen innerhalb einer Zeichenkette nach einem vorgegeben Schema vertauscht werden: Der zu bearbeitende String ist dabei erstes Argument. Danach folgen die Ziel- und die Quelltable mit Zeichen: Im vorgegebenen String wird dann jedes Zeichen, das in der Quelltable vorhanden ist, durch das Zeichen, das an der korrespondierenden Stelle in der Zieltabelle steht, ersetzt. Ist die Zieltabelle kürzer als die Quelltable, so werden die in Frage kommenden Zeichen durch ein Leerzeichen ersetzt, wenn nicht als vierter Parameter ein spezielles Füllzeichen angegeben wird.

Wird 'Translate()' nur mit einem Quellstring aufgerufen, so übersetzt es alle Buchstaben in der übergebenen Zeichenkette in Großbuchstaben - genau wie die Funktion 'Upper()', wobei letzere etwas schneller ist. Allerdings stören sich beide Funktionen an nationalen Sonderzeichen wie z.B. Umlauten: Wer solche Strings in Großschrift wandeln möchte, der muß sich anders behelfen.

```
/* Rot'13 */
```

```
src = Xrange("a","z")  
dest = Right(src,13) || Left(src,13)
```

```
src = src || Upper(src)  
dest = dest || Upper(dest)
```

```
do until Eof(stdin)  
Call Writeln(stdout,Translate(Readln(stdin),dest,src))  
end
```

```
exit
```

Die 'Cäsarische Verschlüsselung' ist eine klassische Codierung, bei dem der n-te Buchstabe durch den (n+13)-ten Buchstaben ersetzt wird, wobei jenseits der Position des 'z' das Alphabet wieder bei 'a' beginnt. Dieses Beispiel ist als Filter konzipiert, d.h., das Programm liest von der Standard-Eingabe, und schreibt auf die Standard-Ausgabe.

Eine weitere Funktion zur Manipulation von ganzen Zeichenketten ist 'Reverse()': Wie der Name schon vermuten läßt, wird mit dieser Funktion eine ganze Zeichenkette einfach umgekehrt. 'Center()' zentriert einen als erstes Argument angegebenen String in einen Zielstring der als zweites Argument angegebenen Länge. Ist diese Länge größer als die tatsächliche Länge des Quellstrings, so wird an den Rändern mit Leerzeichen oder einem optional angebbaren Zeichen aufgefüllt, ansonsten werden Anfang und Ende des Strings so beschnitten, daß ein Zielstring der geforderten Länge entsteht.

```
/* Convert */
```

```
/*
```

Syntax: Convert <file> <I|A>

Wandelt Sonderzeichen ins <I>BM oder <A>miga-format

```
*/
```

```
sum = 0
```

```
parse arg filename jobcode .
```

```
jobcode = Upper(Left(jobcode,1))
```

```
if Words(Arg(1)) ~= 2 then filename = '?'
```

```
if Index('AI',jobcode) = 0 then filename = '?'
```

```
src = 'öÖäÄüÜß'
```

```
koe = d2c(148); Oe = d2c(153); kae = d2c(132); Ae = d2c(142)
```

```
kue = d2c(129); Ue = d2c(154); kss = 'e1'x
```

```
des = koe||Oe||kae||Ae||kue||Ue||kss
```

```
LF = '0d0a'x
xten = '.ibm'
```

```
if jobcode = 'A' then do
help = des; des = src; src = help; LF = '0a'x; xten = '.amg'
end
```

```
if filename = '?' then Exit(syntax_display())
if Open('infile',filename,'R') = 0 then Exit(no_file())
if Open('outfile',filename||xten,'W') = 0 then Exit(no_outfile())
```

```
do until Eof('infile')
zeile = Strip(Translate(Readln('infile'),des,src),T,'0d'x)
Call Writech('outfile',zeile||LF)
end
```

```
exit
```

```
Syntax_display:
do i = 3 TO 6; say Sourceline(i); end
Return(0)
```

```
No_file:
say 'Fehler: Kann Quelldatei nicht öffnen!' '0a'x
Return(20)
```

```
No_outfile:
say 'Fehler: Kann Ausgabedatei nicht öffnen!' '0a'x
Return(20)
```

Das Utility Convert konvertiert ASCII-Texte zwischen IBM- und Amiga-Format. Hier finden Sie eine Demonstration aller wesentlichen String Funktionen.

PAG Sandini

Funktionsbibliotheken

ARexx kann über geeignete Funktionsbibliotheken - die REXX Function Hosts - jederzeit um neue Funktionen erweitert werden: So bietet beispielsweise die REXXSupport.Library nützliche Amiga-spezifische Funktionen wie z.B. direkten Speicherzugriff, Funktionen zum Direktzugriff auf Dateien oder zur Verwaltung eines eigenen Message-Ports.

Damit ARexx die Funktionsbibliotheken nutzen kann, müssen Sie zunächst einmal mit der Funktion 'AddLib()' eingebunden werden, wobei als Argumente neben dem Namen der Bibliothek auch Priorität, Offset und Mindest-Version anzugeben sind. Danach können die Funktionen der Funktionsbibliothek wie normale ARexx-Funktionen genutzt werden.

/* Öffne eine REXX Function Library */

/* Aufruf:

**** OK = OpenLib(<name>,<prio>,<offset>,<rev>)***

****/***

**** Es_müssen_vier Argumente angegeben werden */***

Wenn die Library nicht schon geladen ist, so */

versucht OpenLib() sie zu öffnen. Sonst wird */

:False> als Fehler zurückgegeben. */

ARG() = 4 &

dex(Show(libs),Arg(1)) = 0 |

dLib(Arg(1),Arg(2),Arg(3),Arg(4)) ,

EN Return(1)

E Return(0)

**funktion 'OpenLib()' öffnet REXX Function Libraries zuverlässiger als
ingebaute 'AddLib()'.*

*Die Dilemma ergibt sich aus der Verwendung von 'AddLib()' allerdings,
dieser Funktion liefert auch einen Fehler zurück, wenn eine bereits geöffnete*

nete Bibliothek zum zweiten mal geöffnet werden soll. Man kann den Fehler einfach ignorieren, indem man 'AddLib()' mit 'CALL' aufruft, doch dann vergibt man sich auch die Chance, einen Fehler abzufangen, wenn die erforderliche Bibliothek nicht geöffnet werden kann.

Unser nächstes Beispiel liefert einen brauchbaren Kompromiß: Das ARexx-Programm 'OpenLib.rexx' kann als Funktion innerhalb eines anderen ARexx-Programms genutzt werden, und dort 'AddLib()' ersetzen.

```
/* Einfaches 'LIST' für ARexx */
```

```
/* Verzeichnisname als Argument */  
parse arg dir .
```

```
NL = '0a'x /* Zeilenvorschub */  
bytes = 0 /* Byte-Zähler */
```

```
/* Prüfen, ob Argument ein existierendes Verzeichnis ist */  
if ~Exists(dir) & (Left(StateF(dir),3) ~= 'DIR') then do  
say 'Verzeichnis' dir 'existiert nicht!'  
exit(5)  
end
```

```
if ~OpenLib('rexxsupport.library',0,-30,0) then do  
say 'Kann REXXSupport.library nicht öffnen!'  
exit(5)  
end
```

```
/* Dateien im Verzeichnis in String einlesen */  
liste = ShowDir(dir,'file')
```

```
/* Kopfzeile ausgeben */  
say NL||'Dateien im Verzeichnis'  
dir say Copies('=',Length(dir)+23) NL
```

```
/* Unterverzeichnisnamen um Slash erweitern */  
if Right(dir,1) ~= ':' then  
dir = dir || '/'
```

```
do while liste ~= ''
/* Ein Wort nach dem anderen aus der Liste holen */
parse var liste file liste
/* Status für Datei im Verzeichnis dir abfragen */
status = StateF(dir||file)
/* Informationen aus Status-String lesen */
parse var status x size x prot .
/* Gesamtgröße hochzählen */
bytes = bytes + size
/* Dateinamen formatieren */
file = Overlay(file,Copies(' ',20))
/* Bytezahl formatieren */
size = Right(Copies(' ',6) || size,6)
say file prot ' (' size 'bytes)'
end

say NL||'=== '
say 'Insgesamt' bytes 'bytes belegt' NL

exit
```

Ein einfaches List-ähnliches Programm nutzt Funktionen der REXX-Support.library, um sich Informationen über Files zu beschaffen. Als Argument wird der Name eines Verzeichnisses angegeben, und als Resultat werden alle Files in dem Verzeichnis aufgelistet, und ihre Filegrößen addiert.

Befehlsumgebungen

Ein wichtiges Einsatzgebiet von ARexx haben wir bis jetzt noch nicht kennengelernt: Den Einsatz von ARexx als Makrosprache. Bevor wir zu den technischen Details kommen, wollen wir uns kurz mit der nötigen Theorie auseinander setzen: ARexx 'kennt' nicht nur seine eigenen Befehle, sondern auch diejenigen, die von der jeweiligen Befehlsumgebung zur Verfügung gestellt werden. Zudem kann auch jede gerade aktive Befehlsumgebung mit dem 'ADDRESS'-Befehl angewählt werden, womit dann deren Funktionen zur Verfügung stehen. Der Name der jeweiligen Befehlsumgebung ist der des

damit verbundenen AReXX-Ports: So lautet der Portname des CygnusEd 'REXX_CED', während der aktuelle Kommandointerpreter als 'COM-MAND' angesprochen wird.

Je nach Implementation des AReXX-Port liefern Funktionen einer anderen Befehlsumgebung Ergebnisse in der Variablen 'RESULT' zurück: Damit dieser Mechanismus einwandfrei funktioniert, muß er mit 'options result' aktiviert werden.

/* Fakultät im CED */

options results

GETNUMBER 5 "Bitte eine Zahl:"

Zahl = +RESULT

TEXT 'Die Fakultät von' Zahl 'ist' Fak(Zahl)||'.'

exit

PAG Sandini

Fak: procedure

x = arg(1)

if x > 1 then

x = x * Fak(x-1)

return x

Das Fakultäten-Programm nutzt jetzt den CygnusEd zur Ein- und Ausgabe. Es muß dazu als Makro-Programm im Special-Menü installiert sein. Mit 'GETNUMBER' wird ein CED-Requester erzeugt, und 'TEXT' fügt eine Zeichenkette an der aktuellen Cursor-Position in den Text ein.

Wenn ein AReXX-Programm aus einer Applikation heraus gestartet wird, dann ist die Applikation automatisch die aktuelle Befehlsumgebung des Programms: Es kann jedoch jederzeit auch in eine andere Befehlsumgebung gewechselt werden, so daß ein Programm effektiv ein anderes Programm steuern kann.

Falls eine Applikation über ihrem AReXX-Port einen Befehl anbietet, der zufälligerweise den gleichen Namen wie einer der "echten" AReXX-Befehle hat,

so muß der Anwender seinen Befehl in Anführungszeichen packen, so daß ARexx ihn nicht mehr als eingebauten Befehl erkennt, und ihn an die Applikation zur Auswertung weiterreicht.

Applikations-Programmieren in ARexx ist also keinesfalls schwierig, wenn die Entwickler ein durchdachtes Interface vorgegeben haben. Glücklicherweise trifft das heutzutage auf die meisten ARexx-fähigen Programme zu.

PAG Sandini

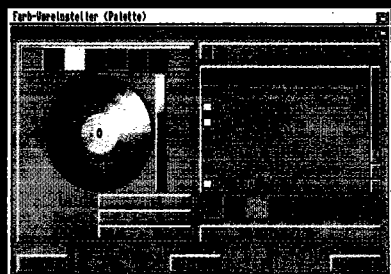
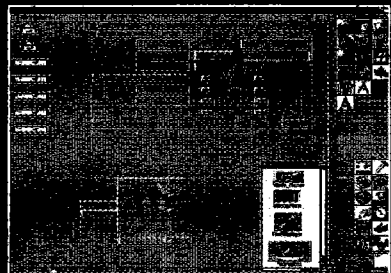
PAG Sandini

PAG Sandini

PAG Sandini

AMIGA

DOS



Autor: Anton Mückl

Daß sich die Maus als Eingabegerät bei Computern etablieren müssen sogar die Kritiker heute zugeben. Bietet doch die Benutzeroberfläche einen leichten und damit schnellen Zugang zur Welt des Computers. Auch beim AMIGA erleichtert die neue Benutzeroberfläche den ersten Arbeiten mit dem Computer. Nicht alle Aufgaben lassen sich mit einer Benutzeroberfläche lösen und so wurde auch dem AMIGA die Möglichkeit einer Bedienung per Tastatur mitgegeben.

Leider ist vielen AMIGA-Modellen, so auch dem A 1200, kein Handbuch über „AMIGA DOS“ beigelegt. In dem ein Benutzer die einzelnen Befehle enthalten gewesen wäre. Der Ein- und Aufsteiger zum AMIGA DOS Version 3.0 bekommt so die Chance, die Shell als Alternative der Workbench zu erlernen. Diese Möglichkeit will dieses Buch geben. Es ist für den Internen, für Neueinsteiger oder Fortgeschrittene geeignet.

Aus dem Inhalt:

- Die Menüfunktionen der Workbench
- Erstellung von Batch-Dateien
- Umgang mit der Shell
- Die AMIGA DOS-Befehle
- Texteditor MiEmaos
- AMIGA DOS-Fehlermeldungen
- ARexx für Einsteiger

Die beiliegende Diskette enthält viele nützliche Programme für die Workbench und Shell.

Systemvoraussetzungen für beiliegende Diskette:
AMIGA mit mind. 1 MB RAM und Kickstart/Workbench



DM 39,90
ÖS 329,-
Sfr 39,90



media
verlagsgesellschaft mbH

Vertrieb Österreich:

High-Tech

Best.-Nr. 1710